Exact and approximate inference in graphical models: variable elimination and beyond

Nathalie Dubois Peyrard, Simon de Givry, Alain Franc, Stephane Robin, Régis Sabbadin, Thomas Schiex, Matthieu Vignes

Exact and approximate inference in graphical models: variable elimination and beyond

Nathalie Peyrard[a], Simon de Givry[a], Alain Franc[b], Stéphane Robin[c,d], Régis Sabbadin[a], Thomas Schiex[a], Matthieu Vignes[a]

[a] INRA UR 875, Unité de Mathématiques et Informatique Appliquées, Chemin de Borde Rouge, 31326 Castanet-Tolosan, France
[b] INRA UMR 1202, Biodiversité, Gènes et Communautés, 69, route d'Arcachon, Pierroton, 33612 Cestas Cedex, France
[c] AgroParisTech, UMR 518 MIA, Paris 5e, France
[d] INRA, UM R518 MIA, Paris 5e, France

June 30, 2015

Abstract

Probabilistic graphical models offer a powerful framework to account for the dependence structure between variables, which can be represented as a graph. The dependence between variables may render inference tasks such as computing normalizing constant, marginalization or optimization intractable. The objective of this paper is to review techniques exploiting the graph structure for exact inference borrowed from optimization and computer science. They are not yet standard in the statistician toolkit, and we specify under which conditions they are efficient in practice. They are built on the principle of variable elimination whose complexity is dictated in an intricate way by the order in which variables are eliminated in the graph. The so-called treewidth of the graph characterizes this algorithmic complexity: low-treewidth graphs can be processed efficiently. Algorithmic solutions derived from variable elimination and the notion of treewidth are illustrated on problems of treewidth computation and inference in challenging benchmarks from optimization competitions. We also review how efficient techniques for approximate inference such as loopy belief propagation and variational approaches can be linked to variable elimination and we illustrate them in the context of Expectation-Maximisation procedures for parameter estimation in coupled Hidden Markov Models.

Keywords: graphical model, computational inference, treewidth, message passing, variational approximations

1 Introduction

Most real complex systems are made up or modeled by elementary objects that locally interact with each other. Graphical models (Bishop, 2006; Koller and Friedman, 2009; Murphy, 2012) are formed by variables linked to each other by stochastic relationships. They enable to model dependencies in possibly high-dimensional heterogeneous data and to capture uncertainty. Graphical models have been applied in a wide range of areas like image analysis, speech recognition, bioinformatics, ecology to name a few.

In real applications a large number of random variables with a complex dependency structure are involved. As a consequence, inference tasks such as the calculation of a normalization constant, a marginal distribution or the mode of the joint distribution are challenging. Three main approaches exist to evaluate such quantities for a given distribution p defining a graphical model: (a) compute them in an exact manner; (b) use a stochastic algorithm to sample from the distribution p to get (unbiased) estimates; (c) derive an approximation of p for which the exact calculation is possible. Even if appealing, exact computation on p often leads to very time and memory consuming procedures, since the number of elements to store or elementary operations to perform increase exponentially with n the number of random variables. The second approach is probably the most widely used by statisticians and modelers. Stochastic algorithms such as Monte-Carlo Markov Chains (MCMC, Robert and Casella, 2004), Gibbs sampling (Casella and George, 1992) and particle filtering (Gordon and Smith, 1993) have become standard tools in many fields of application using statistical models. The last approach includes variational approximation techniques (Wainwright and Jordan, 2008), which are starting to become common practice in computational statistics. In essence, approaches of type (b) provide an approximate answer to an exact problem whereas approaches of type (c) provide an exact answer to an approximate problem.

In this paper we focus on approaches of type (a) and (c), and we will review techniques for exact or approximate inference in graphical models borrowed from both optimization and computer science. They are computationally efficient, yet not standard in the statistician toolkit. Our purpose is to show that the characterization of the structure of the graph G associated to a graphical model (precise definitions are given in Section 2) enables both to determine if the exact calculation of the quantities of interest (marginal distribution, normalization constant, mode) can be implemented efficiently and to derive a class of operational algorithms. When the answer is no, the same analysis enables to design algorithms to compute an approximation of the desired quantities for which an acceptable complexity can be obtained.

The central algorithmic tool is the variable elimination concept (Bertelé and Brioshi, 1972). In Section 3 we adopt a unified algebraic presentation of the different inference tasks (marginalization, normalizing constant or mode evaluation) to emphasize that all of them can be solved as particular cases of variable elimination. This implies that if variable elimination is efficient for one task it will also be efficient for the other ones. The key ingredient to design efficient algorithms based on variable elimination is the clever use of distributivity between algebraic operators. For instance distributivity of the product ($\times$) over the sum ($+$) enables to write $(a \times b) + (a \times c) = a \times (b + c)$ and evaluating the left-hand side of this equality requires two multiplications and one addition while evaluating the right-hand side requires one multiplication and one addition. Similarly since $\max(a + b, a + c) = a + \max(b, c)$ it is more efficient to compute the right-hand side from an algorithmic point of view. Distributivity enables to minimize the number of operations.

Associativity and commutativity are also required and the algebra behind is the semi-ring category (from which some notations will be borrowed). Inference algorithms using the distributivity property have been known and published in the Artificial Intelligence and Machine Learning literature under different names, such as sum-prod, or max-sum (Pearl, 1988; Bishop, 2006) and are examples of variable elimination.

Variable elimination relies on the choice of an order of elimination of the variables (either by marginalization or by maximization). This corresponds to the ordering calculations are performed when applying distributivity. The topology of the graph G provides key information to organize the calculations for an optimal use of distributivity, i.e. to minimize the number of elementary operations to perform. For example, when the graph is a tree, the most efficient elimination order corresponds to eliminating recursively the vertices of degree one, starting from the leaves towards the root. For an arbitrary graph, the notion of an optimal elimination order for inference in a graphical model is closely linked to the notion of treewidth of the associated graph G. We will see in Section 3 the reason why inference algorithms based on variable elimination with the best elimination order are of complexity linear in n but exponential in the treewidth. Therefore treewidth is the key characterization of G to determine if exact inference is possible in practice or not.

The concept of treewidth has been proposed in parallel in computer science (Bodlaender, 1994) and in discrete mathematics and graph minor theory (see Robertson and Seymour, 1986; Lovász, 2005). Discrete mathematics existence theorems (Robertson and Seymour, 1986) establish that there exists an algorithm for computing the treewidth of any graph with complexity polynomial in n (but exponential in the treewidth), and even the degree of the polynomial is given. However this result does not tell how to derive and implement the algorithm, apart from some specific cases (as trees, chordal graphs, and series-parallel graphs, see Duffin (1965). So we will also present in Section 4 several state-of-the-art algorithms for approximate evaluation of the treewidth and illustrate their behavior on benchmarks borrowed from optimization competitions.

Variable elimination has also lead to message passing algorithms (Pearl, 1988) which are now common tools in computer science or machine learning. More recently these algorithms have been reinterpreted as re-parametrization tools (Koller and Friedman, 2009). We will explain in Section 5 how re-parametrization can be used as a pre-processing tool to transform the original graphical model into an equivalent one for which inference may been simpler. Message passing is not the only way to perform re-parametrization and we will discuss alternative efficient algorithms that have been proposed in the context of constraint satisfaction problems (CSP, see (Rossi et al., 2006)) and that have not yet been exploited in the context of graphical models.

As emphasized above, efficient exact inference algorithms can only be designed for graphical models with limited treewidth (much less than the number of vertices), which is a far from being the general case. But the principle of variable elimination and message passing for a tree can still be applied to any graph leading then to heuristic inference algorithms. The most famous heuristics is the Loopy Belief Propagation algorithm (Kschischang et al., 2001) We recall in Section 6 the result that establishes LBP as a variational approximation method. Variational methods rely on the choice of a distribution which renders inference easier, to approximate the original complex graphical model p. The approximate distribution is chosen within a class of models for which efficient inference algorithms exist, that is models with small treewidth (0, 1 or 2 in practice). We review some of the standard choices and we illustrate on the problem of parameter estimation in coupled Hidden Markov Model (Ghahramani and Jordan, 1997) how variational methods have

been applied in practice with different approximate distributions, each of them corresponding to a different underlying treewidth (Section 7).

2 Graphical Models

2.1 Models definition

Consider a a stochastic system defined by a set of random variables $X = (X_1, \ldots, X_n)$. Each variable X_i takes values in Λ_i. Then, a realization of X is a set $x = (x_1, \ldots, x_n)$, with $x_i \in \Lambda_i$. The set of all possible realizations is called the state space, and is denoted $\Lambda = \prod_{i=1}^{n} \Lambda_i$. If A is a subset of $V = \{1, \ldots, n\}$, X_A, x_A and Λ_A are respectively the subset of random variables $\{X_i, i \in A\}$, the set of realizations $\{x_i, i \in A\}$ and the state space of X_A. If p is the joint probability distribution of X on Λ, we denote

$$\forall\, x \in \Lambda, \quad p(x) = p(X = x).$$

Note that we focus here on discrete variables (we will discuss inference in the case of continuous variables on examples in Section 8). A joint distribution p on Λ is said to be a *probabilistic graphical model* (Lauritzen, 1996; Bishop, 2006; Koller and Friedman, 2009) indexed on a set $\mathcal{B}$ of parts of V if there exists a set $\Psi = \{\psi_B\}_{B \in \mathcal{B}}$ of maps from Λ_B to $\mathbb{R}^+$, called *potential functions*, indexed by $\mathcal{B}$ such that p can be expressed in the following factorized form:

$$p(x) = \frac{1}{Z} \prod_{B \in \mathcal{B}} \psi_B(x_B), \tag{1}$$

where $Z = \sum_{x \in \Lambda} \prod_{B \in \mathcal{B}} \psi_B(x_B)$ is the normalizing constant, also called partition function. The elements $B \in \mathcal{B}$ are the scopes of the potential functions and $|B|$ is the arity of the potential function ψ_B. The set of scopes of all the potential functions involving variable X_i is denoted $\mathcal{B}_i = \bigcup_{\substack{B \in \mathcal{B} \\ B \ni i}} \{B\}$.

One desirable property of graphical models is that of Markov local independence: if $p(X = x)$ can be expressed as (1) then a variable X_i is (stochastically) independent of all others in X conditionally to the set of variables $X_{(\cup_{B \in \mathcal{B}_i} B) \setminus i}$. This set is called the Markov blanket of X_i, or its neighborhood. We will denote it N_i. These conditional independences can be represented graphically, by a graph with one vertex per variable in X. The question of encoding the independence properties associated with a given distribution into a graph structure is well described in (Koller and Friedman, 2009), and we will not discuss it here. We will consider the classical graph $G = (V, E)$ associated to a decomposition of the form (1) where an edge is drawn between two vertices i and j if there exists $B \in \mathcal{B}$ such that i and j are in B. Such a representation of a graphical model is actually not as rich as the representation (1). For instance, if $n = 3$, the two cases $\mathcal{B} = \{\{1, 2, 3\}\}$ and $\mathcal{B} = \{\{1, 2\}, \{2, 3\}, \{3, 1\}\}$ are represented by the same graph G, namely a clique of size 3. The factor graph representation goes beyond this limit: this graphical representation is a bipartite graph with one vertex per potential function and one vertex per variable. edges are only between functions and variables. An edge is present between a function vertex (also called factor vertex)

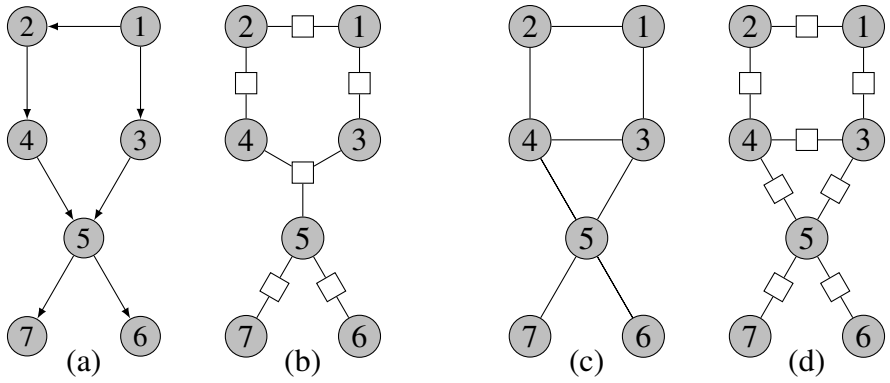

Figure 1: From left to right: (a) Graphical representation of a directed graphical model where potential functions define the conditional probability of each variable given its parents values; (b) The corresponding factor graph where every potential function is represented as a factor (square vertex) connected to the variables that are involved in it; (c) Graphical representation of an undirected graphical model. It is impossible from this graph to distinguish between a graphical model defined by a unique potential function on vertices 3, 4 and 5 from a model defined by 3 pairwise potential functions over each pair $(3, 4)$, $(3, 5)$ and $(4, 5)$; (d) The corresponding factor graph, which unambiguously defines these potential functions (here three pairwise potential functions)

and a variable vertex if the variable is in the scope of the potential function. Figure 1 displays examples of the two graphical representations.

There exists several families of probabilistic graphical models (Koller and Friedman, 2009; Murphy, 2012). They can be grouped into directed and undirected ones. The most classical directed framework is that of *Bayesian network* (Pearl, 1988; Jensen and Nielsen, 2007). In a Bayesian network, an edge is directed from a parent vertex to a child vertex and potential functions are conditional probabilities of a variable given its parents in the graph (see Figure 1 (a)). In such models, trivially $Z = 1$. Undirected probabilistic graphical models (see Figure 1 (c)) are equivalent to *Markov Random Fields* (Li, 2001) as soon as the potential functions are in $\mathbb{R}^{+*}$. In a Markov random field (MRF), a potential function is not necessarily a probability distribution: ψ_B is not required to be normalized.

Deterministic Graphical models. Although the terminology of 'Graphical Models' is often used to refer to stochastic graphical models, the idea of describing a joint distribution on a set of variables through local functions has also been used in Artificial Intelligence to concisely describe Boolean functions or cost functions, with no normalization constraint. In a graphical model with only Boolean (0/1) potential functions, each potential function describes a constraint between variables. If the potential function takes value 1, the corresponding realization is said to satisfy the constraint. The graphical model is known as a 'Constraint Network'. It describes a joint Boolean function on all variables that takes value 1 if and only if all constraints are satisfied. The problem of finding a realization that satisfies all the constraints, called a solution of the network, is the 'Constraint Satisfaction Problem' (CSP) (Rossi et al., 2006). This framework is used to model and solve combinatorial optimization problems and there is a variety of software tools to solve it.

When variables are Boolean too and when the Boolean functions are described as disjunctions of variables or of their negation, the CSP reduces to the 'Boolean Satisfiability' problem (or SAT), the seminal NP-complete problem (Cook, 1971).

CSP have been extended to describe joint cost functions, decomposed as a sum of local cost functions in the 'Weighted Constraint Network' (Rossi et al., 2006) or 'Cost Function Network'. In this case, potential functions take finite or infinite integer or rational values: infinity enables to express hard constraints while finite values encode costs for unsatisfied soft constraints. The problem of finding a realization of minimum cost is the 'Weighted Constraint Satisfaction Problem' (WCSP), which is also NP-hard. It is easy to observe that any stochastic graphical model can be translated in a weighted constraint network using a simple $-\log(\cdot)$ transformation. With this equivalence, it becomes possible to use exact WCSP resolution algorithms that have been developed in this field for mode evaluation in stochastic graphical model.

2.2 Inference tasks in probabilistic graphical models

Computations on probabilities and potentials rely on two fundamental types of operations. Firstly, multiplication (or addition in the $\log$ domain) is used to *combine* potentials to define a joint potential distribution. Secondly, sum or $\max$/$\min$ can be used to *eliminate* variables and compute marginals or modes of the joint distribution on subsets of variables. The precise identity of these two basic operations is not crucial for the inference algorithms considered in this paper. We denote as $\odot$ the combination operator and as $\oplus$ the elimination operator. The algorithms just require that $(\mathbb{R}^+, \oplus, \odot)$ defines a commutative semi-ring. Specifically, the semi-ring algebra offers distributivity: $(a \odot b) \oplus (a \odot c) = a \odot (b \oplus c)$. This corresponds to distributivity of product over sum since $(a \times b) + (a \times c) = a \times (b + c)$ or distributivity of $\max$ over sum since $\max(a + b, a + c) = a + \max(b, c)$, or again distributivity of $\max$ over product since $\max(a \times b, a \times c) = a \times (\max(b, c))$. These two abstract operators can be defined to be applied to potential functions, as follows:

Combine operator: the combination of two potential functions ψ_A and ψ_B is a new function $\psi_A \odot \psi_B$, from $\Lambda_{A \cup B}$ to $\mathbb{R}^+$ defined as $\psi_A \odot \psi_B(x_{A \cup B}) = \psi_A(x_A) \odot \psi_B(x_B)$.

Elimination operator: the elimination of variable $X_i, i \in B$ from a potential function ψ_B is a new function $(\oplus_{x_i} \psi_B)$ from $\Lambda_{B \setminus \{i\}}$ to $\mathbb{R}^+$ defined as $(\oplus_{x_i} \psi_B)(x_{B \setminus \{i\}}) = \oplus_{x_i}(\psi_B(x_{B \setminus \{i\}}, x_i))$. For $\oplus = +$, $(\oplus_{x_i} \psi_B)(x_{B \setminus \{i\}})$ represents $\sum_{x_i} \psi_B(x_{B \setminus \{i\}}, x_i)$.

We can now describe classical counting and optimization tasks in graphical models in terms of these two operators. For simplicity, we denote by $\oplus_{x_B}$, where $B \subset V$ a sequence of eliminations $\oplus_{x_i}$ for all $i \in B$, the result being insensitive to the order in a commutative semi-ring. Similarly, $\odot_{B \in \mathcal{B}}$ represents the successive combination of all potential functions ψ_B such that $B \in \mathcal{B}$.

Counting tasks. Under this name we group all tasks that involve summing over the state space of a subset of variables in X. This includes the computation of the partition function Z or of any

marginal distribution, as well as entropy evaluation. For $A \subset V$ and $\bar{A} = V \setminus A$, the marginal distribution p_A of X_A associated to the joint distribution p is defined as:

$$p_A(x_A) = \sum_{x_{\bar{A}} \in \Lambda_{\bar{A}}} p\left(x_A, x_{\bar{A}}\right) = \frac{1}{Z} \sum_{x_{\bar{A}} \in \Lambda_{\bar{A}}} \prod_{B \in \mathcal{B}} \psi_B(x_B) \tag{2}$$

The function p_A then satisfies:

$$p_A \odot Z = p_A \odot \Big(\bigoplus_{x_V} \big(\bigodot_{B \in \mathcal{B}} \psi_B \big) \Big) = \Big(\bigoplus_{x_{\bar{A}}} \big(\bigodot_{B \in \mathcal{B}} \psi_B \big) \Big)$$

where $\odot$ combines functions using $\times$ and $\oplus$ eliminates variables using $+$.

Marginal evaluation is also interesting in the case where some variables are observed. If the values of some variables x_O ($O \subset V$) have been observed, we can compute the marginal conditional distribution by restricting the domains of variables X_O to the observed value.

The entropy H of a probabilistic graphical model p is defined as

$$H(p) = -E[\ln(p(x))], \tag{3}$$

where $E[\cdot]$ denotes the mathematical expectation. In the case of a graphical model, by linearity of the expectation, the entropy is equal to

$$H(p) = \ln(Z) - \sum_{B \in \mathcal{B}} \sum_{x_B \in \Lambda_B} p(x_B) \ln(\psi_B(x_B)).$$

This expression is an alternation of use of $\odot$ and $\oplus$ operators (for $p(x_B)$ evaluation, for each B and x_B).

Optimization task An optimization task in a graphical model corresponds to the evaluation of the most probable state x^* of the random vector X, defined as

$$x^* = \arg\max_{x \in \Lambda} p(x) = \arg\max_{x \in \Lambda} \prod_{B \in \mathcal{B}} \psi_B(x_B) = \arg\max \sum_{B \in \mathcal{B}} \ln \psi_B(x_B). \tag{4}$$

The maximum itself is $\bigoplus_{x_V} \bigodot_{B \in \mathcal{B}} \ln \psi_B(x_B)$ with $\oplus$ set to $\max$ and $\odot$ to $+$. The computation of the mode x^* does not require the computation of the normalizing constant Z, however computing the mode probability $p(x^*)$ does.

Therefore counting and optimization tasks can be interpreted as two instantiations of the same computational task expressed in terms of combination and elimination operators, namely $\bigoplus_{x_A} \bigodot_{B \in \mathcal{B}} \psi_B$ where $A \subset V$. When the combination operator $\odot$ and the elimination operator $\oplus$ are respectively set to $\times$ and $+$, this computational problem is known as a sum-product problem in the Artificial Intelligence literature (Pearl, 1988),(Bishop, 2006, chapter 8). When $\oplus$ is set to $\max$ and $\odot$ to the sum operator it is a max-sum problem (Bishop, 2006, chapter 8).

We will see in Section 3 that there exists an exact algorithm solving this general task that exploits the distributivity of the combination and elimination operators to perform operations in a smart order. From this generic algorithm, known as variable elimination (Bertelé and Brioshi, 1972) or bucket elimination (Dechter, 1999), one can deduce exact algorithms to solve counting and optimization tasks in a graphical model, by instantiating the operators $\oplus$ and $\odot$.

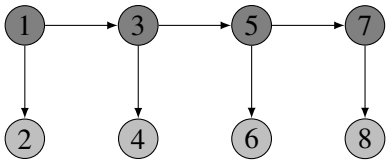

Figure 2: Graphical representation of a HMM. Hidden variables correspond to vertices 1, 3, 5, 7, and observed variables to vertices 2, 4, 6, 8.

Deterministic Graphical models : the Constraint Satisfaction Problem is a $\vee$-$\wedge$ problem as it can can be defined using $\vee$ (logical 'or') as the elimination operator and $\wedge$ (logical 'and') as the combination operator over Booleans. The weighted CSP is a $\min$-$+$ as it uses $\min$ as the elimination operator and $+$ (or bounded variants of $+$) as the combination operator. Several other variants exist (Rossi et al., 2006), including generic algebraic variants (Schiex et al., 1995; Bistarelli et al., 1997; Cooper, 2004; Pralet et al., 2007; Kohlas, 2003).

3 Variable elimination for exact inference

We describe now the principle of variable elimination. We first recall the Viterbi algorithm for Hidden Markov Chains, a classical example of variable elimination for optimization. Then we formally describe the variable elimination procedure in the general graphical model framework. The key element is the choice of an ordering for the sequential elimination of the variables. It is closely linked to the notion of treewidth of the graphical representation of the graphical model. As it will be shown, the complexity of the variable elimination is fully characterized by this notion. Conversely, the treewidth can be bounded from above from a given variable elimination scheme.

3.1 An example: hidden Markov chain models

As an introduction to exact inference on graphical models by variable elimination, we consider a well studied stochastic process: the discrete Hidden Markov Chain model (HMC).

A HMC (Figure 2) is defined by two sequences of random variables O and H of the same length, T. A realization $o = (o_1, \dots o_T)$ of the variables $O = (O_1, \dots O_T)$ is observed, while the states of variables $H = (H_1, \dots H_T)$ are unknown. In the HMC model the assumption is made that O_i is independent of $H_{V\setminus\{i\}}$ and $O_{V\setminus\{i\}}$ given the hidden variable H_i. These independences are modeled by pairwise potential functions $\psi_{H_i,O_i}, \forall\ 1 \leq i \leq T$. Furthermore, hidden variable H_i is independent of $H_1, \dots, H_{i-2}$ and $O_1, \dots, O_{i-1}$ given the hidden variable H_{i-1}. These independences are modeled by pairwise potential functions $\psi_{H_{i-1},H_i}, \forall\ 1 < i \leq T$. Then the model is fully defined by specifying an additional potential function $\psi_{H_1}(h_1)$ to model the initial distribution. In the classical HMC formulation (Rabiner, 1989), these potential functions are normalized conditional probability distributions i.e., $\psi_{H_{i-1},H_i}(h_{i-1}, h_i) = p(H_i = h_i | H_{i-1} = h_{i-1})$, $\psi_{O_i,H_i}(o_i, h_i) = p(O_i = o_i | H_i = h_i)$ and $\psi_{H_1}(h_1) = p(H_1 = h_1)$. As a consequence, the normalizing constant Z is equal to 1, as in any Bayesian network.

A classical inference task for HMC is to identify the most likely value of the variables H given a realization o of the variables O. The problem is to compute $\arg\max_h p(H = h|O = o)$ or equivalently the argument of:

$$\max_{h_1,\ldots,h_T} \left[(\psi_{H_1}(h_1)\psi_{O_1,H_1}(o_1,h_1)) \prod_{i=2}^{T} (\psi_{H_{i-1},H_i}(h_{i-1},h_i)\psi_{O_i,H_i}(o_i,h_i)) \right] \tag{5}$$

The number of possible realizations of H is exponential in T. Nevertheless this optimization problem can be solved in a number of operations linear in T using the well-known Viterbi algorithm (Rabiner, 1989). This algorithm, based on dynamic programming, performs successive eliminations (by maximization) of all hidden variables, starting with H_T, then H_{T-1}, and finishing by H_1, to compute the most likely sequence of hidden variables. By using distributivity between the $\max$ and the product operators, the elimination of variable H_T can be done by rewriting equation (5) as:

$$\max_{h_1,\ldots,h_{T-1}} \Big[\psi_{H_1}(h_1)\psi_{O_1,H_1}(o_1,h_1) \prod_{i=2}^{T-1} \big(\psi_{H_{i-1},H_i}(h_{i-1},h_i)\psi_{O_i,H_i}(o_i,h_i). \underbrace{\max_{h_T} \psi_{H_{T-1},H_T}(h_{T-1},h_T)\psi_{O_T,H_T}(o_T,h_T)}_{\text{New potential function}} \big) \Big]$$

The new potential function created by maximizing on H_T depends only on variable H_{T-1}, so that variables H_T, O_T and potential functions involving them have been removed from the optimization problem. This is a simple application of the general variable elimination algorithm that we describe in the next section.

3.2 General principle of variable elimination

In Section 2, we have seen that counting and optimization tasks can be formalized by the same generic algebraic formulation

$$\bigoplus_{x_A} \Big(\bigodot_{B \in \mathcal{B}} \psi_B \Big) \tag{6}$$

where $A \subset V$.

The trick behind variable elimination (Bertelé and Brioshi, 1972) relies on a clever use of the distributivity property. Indeed, evaluating $(a \odot b) \oplus (a \odot c)$ as $a \odot (b \oplus c)$ requires fewer operations. Since distributivity applies both for counting and optimizing tasks, variable elimination can be applied to both tasks. It also means that if variable elimination is efficient for one task it will also be efficient for the other one. As in the HMC example, the principle of the variable elimination algorithm for counting or optimizing consists in eliminating variables one by one in expression (6).

Elimination of the first variable, say $X_i, i \in A$, is performed by merging all potential functions involving X_i and applying operator $\oplus_{x_i}$ to these potential functions. Using commutativity and associativity equation (6) can be rewritten as follows:

$$\bigoplus_{x_A} \Big(\bigodot_{B \in \mathcal{B}} \psi_B \Big) = \bigoplus_{x_{A \setminus \{i\}}} \bigoplus_{x_i} \left(\Big(\bigodot_{B \in \mathcal{B} \setminus \mathcal{B}_i} \psi_B \Big) \Big(\bigodot_{B \in \mathcal{B}_i} \psi_B \Big) \right)$$

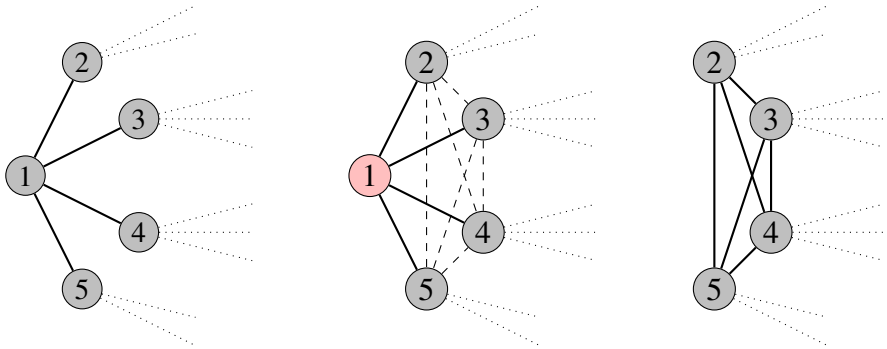

Figure 3: Elimination of variable X_1 replaces the four pairwise potential functions involving variable X_1 with a new potential ψ_{N_1}, involving the four neighbors of vertex 1 in the original graph. The new edges created between these four vertices are called fill-in edges (dashed edges in the middle figure).

Then using distributivity we obtain

$$\bigoplus_{x_A}\Big(\bigodot_{B\in\mathcal{B}}\psi_B\Big)=\bigoplus_{x_{A\setminus\{i\}}}\Big[\Big(\bigodot_{B\in\mathcal{B}\setminus\mathcal{B}_i}\psi_B\Big)\odot\underbrace{\Big(\bigoplus_{x_i}\bigodot_{B\in\mathcal{B}_i}\psi_B\Big)}_{\text{New potential function }\psi_{N_i}}\Big]$$

This shows that the elimination of X_i results in a new graphical model where the variable X_i and the potential functions $\psi_B, B \in \mathcal{B}_i$ have been removed and replaced by a new potential ψ_{N_i} which does not involve X_i, but its neighboring vertices. The graph associated to the new graphical model is similar to the graph of the original model except that vertex X_i has been removed and that the neighbors N_i of X_i are now connected together in a clique. The new edges between the neighbors of X_i are called *fill-in* edges. For instance, when eliminating variable X_1 in the graph of Figure 3 (left), potential functions $\psi_{1,2},\ \psi_{1,3},\ \psi_{1,4}$ and $\psi_{1,5}$ are replaced by $\psi_{2,3,4,5} = \oplus_{x_1}(\psi_{1,2}\odot\psi_{1,3}\odot\psi_{1,4}\odot\psi_{1,4})$. The new graph is shown in Figure 3, right part.

When the first elimination step is applied with $\oplus = +$ and $\odot = \times$, the probability distribution defined by this new graphical model is the marginal distribution $p(x_{V\setminus\{i\}})$ of the original distribution p. The complete elimination can be obtained by successively eliminating all variables in X_A. The result is a graphical model over $X_{V\setminus A}$ which is the marginal distribution $p(x_{V\setminus A})$. When $A = V$, the result is a model with a single constant potential function with value Z.

If instead $\oplus$ is $\max$, and $\odot = \times$ (or $+$ with a log transformation of the potential functions) and $A = V$, the last potential function obtained after elimination of the last variable is equal to the maximum of the non normalized distribution. So evaluating Z or the maximal probability of a graphical model can be both obtained with the same variable elimination algorithm, just changing the meaning of $\oplus$ and $\odot$.
However, if one is interested in the mode itself, an additional simple computation is required. The mode is obtained by induction: if $x^*_{V\setminus\{i\}}$ is the mode of the graphical model obtained after the elimination of the first variable, X_i, then the mode of p can be defined as $(x^*_{V\setminus\{i\}}, x^*_i)$ where x^*_i is a value in Λ_i that maximizes $\odot_{B\in\mathcal{B}}\psi_B(x^*_{V\setminus\{i\}}, x_i)$. This maximization is straightforward to derive because x_i can take only $|\Lambda_i|$ values. $x^*_{V\setminus\{i\}}$ itself is obtained by completing the mode

of the graphical model obtained after elimination the second variable, and so on. We stress here that the procedure requires to keep the intermediary potential functions ψ_{N_i} generated during the successive eliminations.

When eliminating a variable X_i, the task that can be computationally expensive is the computation of the intermediate ψ_{N_i}. It requires to compute the product $\odot_{B \in \mathcal{B}_i} \psi_B(x_B)$ of several potential functions for all elements of $\Lambda_{N_i \cup \{i\}}$, the state space of $X_{N_i \cup \{i\}}$. The time and space complexity of the operation are entirely determined by the cardinality $|N_i|$ of the set of indices N_i. If $K = \max_{j \in V} |\Lambda_j|$ is the maximum domain size of a variable, the time complexity (i.e. number of elementary operations performed) is in $O(K^{|N_i|+1})$ and space complexity (i.e. memory space needed) is in $O(K^{|N_i|})$. Complexity is therefore exponential in $|N_i|$, the number of neighbors of the eliminated variable in the current graphical model. The total complexity of the variable elimination is then exponential in the maximum cardinality $|N_i|$ over all successive eliminations (but linear in n). Because the graphical model changes at elimination each step, this number usually depends on the order in which variables are eliminated.

As a consequence, the prerequisite to apply variable elimination is to decide for an ordering of elimination of the variables. As illustrated in Figure 4 two different orders can lead to two different N_i subsets. The key message is that the choice of the order is crucial/ it dictates the efficiency of the variable elimination procedure. We will now illustrate and formalize this intuition.

3.3 When is variable elimination efficient ?

We can understand why the Viterbi algorithm is an efficient algorithm for mode evaluation in a HMC. The graph associated to a HMC is comb-shaped: the hidden variables form a line and each observed variable is a leaf in the comb (see Figure 2). So it is possible to design an elimination order where the current variable to eliminate has a unique neighbor in the graphical representation of the current model: for instance $O_T > H_T > O_{T-1} > H_{T-1}, \ldots > O_1 > H_1$ (the first eliminated variable is the largest according to this ordering). Following this elimination order, when eliminating a variable using $\oplus$, the resulting graphical model has one fewer vertex than the previous one and *no fill-in edge*. Indeed, the new potential function ψ_{N_i} is a function of a single variable since $|N_i| = 1$.

More generally, variable elimination is very efficient, i.e. leads to intermediate N_i sets of small cardinality, on graphical models whose graph representation is a tree, again because it is always possible to design an elimination order where the current variable to eliminate has only one neighbor in the graphical representation of the current model.

Another situation where variable elimination can be efficient is when the graph associated to the graphical model is chordal (any cycle of length 4 or more has a chord i.e., an edge connecting two non adjacent vertices in the cycle), the size of the largest clique being low. The reason is the following. In Figure 2, new edges are created between neighbors of the eliminated vertex. If this neighborhood is a clique, no new edge is added. A vertex whose neighborhood is a clique is called a simplicial vertex. Chordal graphs have the property that there exists an elimination order of the vertices such that every vertex during elimination process is simplicial. Then, there exists an elimination order such that no fill-in edges are created. Thus, the largest size of N_i is no more than the size of a clique, and is equal to or less than the size of the largest clique in the graph. Let

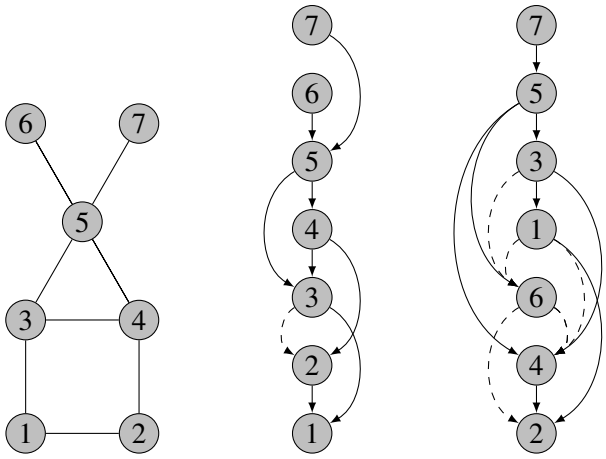

Figure 4: A graph and two elimination orders. Left, the graph; middle, induced graph associated to the elimination order $(7 > 6 > 5 > 4 > 3 > 2 > 1)$. Vertices are eliminated from the largest to the smallest. The maximum size of N_i sets created during elimination is 2 (maximum number of outgoing edges) and only one (dashed) fill-in edge is added when vertex 4 is eliminated; right, induced graph associated to the elimination order $(7 > 5 > 3 > 1 > 6 > 4 > 2)$. The maximum size of N_i sets created during elimination is 3 and 5 (dashed) fill-in edges are used.

us note that a tree is a chordal graph in which all edges and only edges are cliques. Hence, for a tree, simplicial vertices are vertices of degree one. Then, elimination of degree one vertices on a tree is an example of simplicial elimination on a chordal graph.

For arbitrary graphs, if the maximal scope size of the intermediate ψ_{N_i} functions created during variable elimination is too large, then memory and time required for the storage and computation quickly exceed computer capacities. Depending on the chosen elimination order, this maximal scope can be reasonable from a computational point of view, or too large. So again, the choice of the elimination order is crucial.

3.4 The treewidth to characterize variable elimination complexity

The lowest complexity achievable when performing variable elimination is characterized by a parameter called the treewidth of the graph associated to the original graphical model. This concept has been repeatedly discovered and redefined. The treewidth of a graph is sometimes called its induced width (Dechter and Pearl, 1988), its minimum front size (Liu, 1992), its k-tree number (Arnborg, 1985), its dimension (Bertelé and Brioshi, 1972) and is also equal to the min-max clique number of G minus one (Arnborg, 1985). The treewidth is also a key notion in the theory of graph minors (see Robertson and Seymour, 1986; Lovász, 2005).

We insist here on two definitions. The first one from (Bodlaender, 1994) relies on the notion of induced graph and the link between fill-in edges and the intermediate N_i sets created during variable elimination is straightforward. The second (Robertson and Seymour, 1986; Bodlaender, 1994) is the commonly used characterization of the treewidth using so-called tree decompositions, also known as junction trees which are key tools to derive variable elimination algorithms. It underlies the block-by-block elimination procedure described in Section 3.5.

Definition 1 (induced graph) *Let $G = (V, E)$ be a graph defined by a set of vertices indexed on V and a set E of edges. Given an ordering π of the vertices of G, the induced graph G_π^{ind} is obtained as follows. G and G_π^{ind} have same vertices. Then to each edge in E corresponds an oriented edge in G_π^{ind} going from the first of the two nodes according to π toward the second. Then each vertex i of V is considered one after the other following the order defined by π. When vertex i is treated, an oriented edge is created between all pairs of neighbors of i in G that follows i according to π. Again the edge is going from the first of the two nodes according to π toward the second.*

The induced graph G_π^{ind} is also called the *fill graph* of G and the process of computing it is sometimes referred to as "playing the elimination game" on G, as it just simulates elimination on G using the variable ordering π. This graph is chordal (Vandenberghe and Andersen, 2014). It is known that every chordal graph G has at least one vertex ordering π such that $G_\pi^{ind} = G$, called a perfect elimination ordering (Fulkerson and Gross, 1965).

The second notion that enables to define the treewidth is the notion of tree decomposition. Intuitively, a tree decomposition of a graph G organizes the vertices of G in clusters of vertices which are linked by edges such that the graph obtained is a tree. Specific constraints on the way vertices of G are associated to clusters in the decomposition tree are demanded. These contraints ensure properties to tree decomposition usefull for building variable elimination algorithms.

Definition 2 (tree decomposition) *Given a graph $G = (V, E)$, a tree decomposition T is a tree $(\mathcal{C}, E_T)$, where $\mathcal{C} = \{C_1, \ldots, C_l\}$ is a family of subsets of V (called clusters), and E_T is a set of edges between the subsets C_i, satisfying the following properties:*

- *The union of all clusters C_k equals V (each vertex is associated with at least one vertex of T).*
- *For every edge (i, j) in E, there is at least one cluster C_k that contains both i and j.*
- *If clusters C_k and C_l both contain a vertex i of G, then all clusters C_s of T in the (unique) path between C_k and C_l contain i as well: clusters containing vertex i form a connected subset of T. This is known as the running intersection property).*

The concept of tree decomposition is illustrated in Figure 5.

Definition 3 (treewidth) *The two following definitions of the treewidth derived respectively from the notion of induce graph and from that of tree decomposition are equivalent (but this is not trivial to establish):*

- *The treewidth $\mathcal{TW}_\pi(G)$ of a graph G for the ordering π is the maximum number of outgoing edges of a vertex in the induced graph G_π^{ind}. The treewidth $\mathcal{TW}(G)$ of a graph G is the minimum treewidth over all possible orderings π.*
- *The width of a tree decomposition $(\mathcal{C}, E_T)$ is the size of the largest $C_i \in \mathcal{C}$. and the treewidth $\mathcal{TW}(G)$ of a graph is the minimum width among all its tree decompositions.*

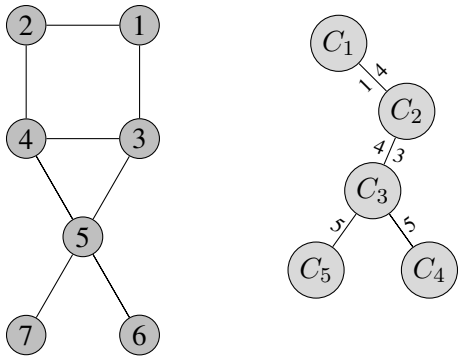

Figure 5: Left: graphical representation of a graphical model. Right: tree decomposition over clusters $C_1 = \{1, 2, 4\},\ C_2 = \{1, 3, 4\},\ C_3 = \{3, 4, 5\},\ C_4 = \{5, 6\}$ and $C_5 = \{5, 7\}$. Each edge between two clusters is labeled by their common variables.

$\mathcal{T}W_\pi(G)$ is exactly the cardinality of the largest set N_i created during variable elimination with elimination order π. For example, in Figure 4, the middle and right graphs are the two induced graphs for two different orderings. $\mathcal{T}W_\pi(G)$ is equal to 2 with the first ordering and to 3 with the second. It is easy to see that in this example $\mathcal{T}W(G) = 2$. The treewidth of the graph of the HMC model, and of any tree is equal to 1.

It has been established that finding a minimum treewidth ordering π for a graph G, finding a minimum treewidth tree decomposition or computing the treewidth of a graph are of equivalent complexity. For an arbitrary graph, computing the treewidth is not an easy task. Section 4 is dedicated to this question, from a theoretical and a practical point of view.

The treewidth is therefore a key indicator to answer the driving subject of this review: will variable elimination be efficient for a given graphical model? For instance, the principle of variable elimination have been applied to the exact computation of the normalizing constant of a Markov random field on a small r by c lattice in (Reeves and Pettitt, 2004). For this regular graph, it is known that the treewidth is equal to $\min(r, c)$. So exact computation through variable elimination is only possible for lattices with a small value for $\min(r, c)$. It is however well beyond computer capacities for real challenging problems in image analysis. In this case variable elimination can be used to define heuristic computational solutions, such as the algorithm of (Friel et al., 2009) which relies on exact computations on small sub-lattices of the original lattice.

3.5 Tree decomposition and block by block elimination

Given a graphical model and a tree decomposition of its graph, a possible alternative to solve counting or optimization tasks is to eliminate variables in successive blocks instead of one after the other. To do so, the block by block elimination procedure (Bertelé and Brioshi, 1972) relies on the tree decomposition characterization of the treewidth. The underlying idea is to apply the variable elimination procedure on the tree decomposition, eliminating one cluster of the tree at each step. First a root cluster $C_r \in \mathcal{C}$ is chosen and used to define an order of elimination of the clusters, by progressing from the leaves toward the roots, such that every eliminated cluster corresponds to a leaf of the current intermediate tree. Then each potential function ψ_B is assigned

to the cluster C_i in $\mathcal{C}$ such that $B \subset C_i$ which is the closest to the root. Sucha cluster always exists from the properties of a tree decomposition and the fact that a potential function is associated to a clique in G). The procedure starts with the elimination of any leaf cluster C_i of T, with parent C_j in T. Let us note $\mathcal{B}(C_i) = \{B \in \mathcal{B}, \psi_B \; assigned \; to \; C_i\}$. Here again, commutativity and distributivity are used to rewrite expression (6) (with $A = V$) as follows:

$$\bigoplus_{x_V} \bigodot_{B \in \mathcal{B}} \psi_B = \bigoplus_{x_{V \setminus (C_i - C_j)}} \Big[\bigodot_{B \in \mathcal{B} \setminus \mathcal{B}(C_i)} \psi_B \odot \underbrace{(\bigoplus_{x_{C_i - C_j}} \bigodot_{B \in \mathcal{B}(C_i)} \psi_B)}_{\text{New potential function}} \Big]$$

Note that only variables with indices in $C_i \setminus C_j \equiv C_i \cap (V \setminus C_j)$ are eliminated, even if it is common to say that the cluster has been eliminated. For instance, for the graph of Figure 5, if the first eliminated cluster is C_1, the new potential function is $\oplus_{x_2} \psi_{1,2}(x_1, x_2)\psi_{2,4}(x_2, x_4)$, it depends only on variables X_1 and X_4. Cluster elimination continues until no cluster is left. The interest of this procedure is that the intermediate potential function created after each cluster elimination may have a scope much smaller than the treewidth, leading to better space complexity (Bertelé and Brioshi, 1972, chapter 4). However, the time complexity is increased.

In summmary, the lowest complexity achievable when performing variable elimination is for elimination orders whose cardinalities of the intermediate N_i sets are lower of equal to the treewidth of G. This treewidth can be determined by considering clusters sizes in tree decompositions of G. Furthermore, a tree decomposition T can be used to build an elimination order and vice versa. Indeed, an elimination order can be defined by using a cluster elimination order based on T and choosing an arbitrary order to eliminate variables with indices in the subsets $C_i \setminus C_j$. Conversely, it is easy to build a tree decomposition from a given vertex ordering π. Since the induced graph G_π^{ind} is chordal, its maximum cliques can be identified in polynomial time. Each such clique defines a cluster C_i of the tree decomposition. Edges of T can be identified as the edges of any maximum spanning tree in the graph with vertices C_i and edges (C_i, C_j) weighed by $|C_i \cap C_j|$.

Deterministic Graphical Models : to our knowledge, the notion of treewidth and its properties have been first identified in combinatorial optimization in (Bertelé and Brioshi, 1972) where it was called "dimension", a graph parameter which has been shown equivalent to the treewidth (Bodlaender, 1998). Variable elimination itself is related to Fourier-Motzkin elimination (Fourier, 1827), a variable elimination algorithm that benefits from the linearity of the handled formulas. Variable elimination has been repeatedly rediscovered, as non-serial dynamic programming (Bertelé and Brioshi, 1972), in the David-Putnam procedure for boolean satisfiability problems (SAT, Davis and Putnam, 1960), as Bucket elimination for the CSP and WCSP (Dechter, 1999), in the Viterbi and Forward-Backward algorithms for HMM (Rabiner, 1989) and many more.

Theree exists other situations where the choice of an elimination order has a deep impact on the complexity of the computations as in Gauss elimination scheme for a system of linear equations, or Choleski factorization of very large sparse matrices, and where equivalence between elimination and decomposition have been used (see Bodlaender et al., 1995).

4 Treewidth computation and approximation

As already mentioned, the complexity of the counting and the optimization tasks on graphical models is heavily linked to the treewidth $\mathcal{TW}(G)$ of the underlying graph G. If one could guess the optimal vertex ordering, π^*, leading to $\mathcal{TW}_{\pi^*}(G) = \mathcal{TW}(G)$, then, one would be able to achieve the "optimal complexity" $O(K^{\mathcal{TW}(G)})n$ for solving exactly these tasks (we recall that K is the maximal domain size of a variable in the graphical model). However, the problem is that one cannot easily evaluate the treewidth of a given graph. The treewidth computation problem is known to be NP-hard (Arnborg et al., 1987).

In the following subsections we provide a short presentation of the state-of-the-art theoretical and experimental results concerning the exact computation of the treewidth of a graph, and the computation of suboptimal vertex orderings providing approximations of the treewidth in the form of an upper bound.

4.1 Exact solution algorithms

Several exponential time exact algorithms have been proposed to compute the treewidth. These algorithms compute the treewidth in time exponential in n. The algorithm with the best complexity bound has been proposed by (Fomin and Villanger, 2012). They provide an exact algorithm for computing the treewidth, which run in time $O(2.6151^n)$ (using polynomial space), or in time $O(1.7549^n)$, using exponential (memory) space.

Since the treewidth of a network can be quite small (compared to n) in practice, there has been a great deal of interest in finding exact algorithms with time complexity exponential in $\mathcal{TW}(G)$ and potentially only polynomial in n. Some of these algorithms even have complexity linear in n (Bodlaender, 1996; Perkovic and Reed, 2000). In Bodlaender (1996), an algorithm is proposed to compute the treewidth (it also provides an associated tree decomposition) of G in time $O(\mathcal{TW}(G)^{O(\mathcal{TW}(G)^3)}n)$. If this algorithm is used to compute the treewidth of graphs in a family of graphs whose treewidth is uniformly bounded, then computing the treewidth would become of time complexity linear in n (however, even for a small bound on the treewidth, the constant can be huge). Moreover, in the general case, there is no way to bound the treewidth a priori.

4.2 Approximation of the treewidth with guarantee

Now, recall that even though crucial, finding a "good" tree decomposition of the graph G is only one element in the computation of quantities of interest in graphical models. If one has to spend more time on finding an optimal vertex ordering than on computing probabilities on the underlying graphical model using an easy-to-compute suboptimal ordering, the utility of exact treewidth computation becomes limited. Therefore, an alternative line of search is to look for algorithms computing a vertex ordering π leading to a suboptimal width, $\mathcal{TW}_\pi(G) \geq \mathcal{TW}(G)$, but more efficient in terms of computational time. When defining such approximation algorithms, one is particularly interested in polynomial time (in n) algorithms, finding a vertex ordering π that approaches the optimal ordering within a constant multiplicative factor : $\mathcal{TW}_\pi(G) \leq \alpha\mathcal{TW}(G), \alpha > 1$.

However, the existence of such constant-factor approximation algorithms is not guaranteed for all NP-hard problems. Some NP-hard problems are even known not to admit polynomial time approximation algorithms. As far as treewidth approximation is concerned, we are in the interesting case where it is not yet known whether or not a polynomial time approximation algorithm does exist (Austrin et al., 2012).

Finally, there have been a variety of proposed algorithms, trading off approximation quality and running time complexity (Robertson and Seymour, 1986; Lagergreen, 1996; Amir, 2010; Bodlaender et al., 2013). Table 1 (extracted from Bodlaender et al., 2013) summarizes the results in terms of approximation guarantee and time complexity for these algorithms.

Algorithm	Approximation guarantee	Time complexity	
		$f(\mathcal{TW}(G))$	$g(n)$
Robertson and Seymour (1986)	$4\mathcal{TW}(G)+3$	$3^{3\mathcal{TW}(G)}$	n^2
Lagergreen (1996)	$8\mathcal{TW}(G)+7$	$2^{\mathcal{TW}(G)\log\mathcal{TW}(G)}$	$n\log^2 n$
Amir (2010)	$3.67\mathcal{TW}(G)$	$2^{3.6982\mathcal{TW}(G)}\mathcal{TW}(G)^3$	n^2
Bodlaender et al. (2013)	$3\mathcal{TW}(G)+4$	$2^{\mathcal{TW}(G)}$	$n\log n$

Table 1: Approximation guarantee and time complexity of state-of-the-art treewidth approximation algorithms. Each algorithm provides a vertex ordering π such that $\mathcal{TW}_\pi(G)$ is upper bounded by the approximation guarantee indicated in column 2. The time complexity of these algorithms is is $O(f(\mathcal{TW}(G)).g(n))$ where n is the number of vertices in G.

The theoretical results about the complexity and approximability of treewidth computation are interesting by the insight they give about the difficulties of finding good, if not optimal, vertex ordering. They are also interesting in that they offer worst-case guarantees, i.e., the approximation quality is guaranteed to be at least that promised by the algorithm. Furthermore, the increase in computation time is also upper-bounded, allowing to get some guarantees that the approximation can be obtained in "reasonable" time.

However, the main drawback of these worst-case based approaches, is that they can be dominated, empirically, by heuristic approaches, on most instances. Indeed, several algorithms, working well in practice, even though without worst-case complexity/quality bounds have been proposed. We describe some of these approaches in the following section.

4.3 Treewidth in practice

A broad class of heuristic approaches is that of greedy algorithms (Bodlaender and Koster, 2010). They use the same iterative approach as the variable elimination algorithm (Section 3) except that they manipulate the graph structure only and do not perform any actual combination/elimination computation. Starting from an empty vertex ordering and an initial graph G, they repeatedly select the next vertex to add in the ordering by locally optimizing one of the following criteria:

- select a vertex with minimum degree in the current graph ;
- select a vertex with minimum number of fill-in edges in the current graph.

After each vertex selection, the current graph is modified by removing the selected vertex and making a clique on its neighbors. The new edges added by this clique are fill-in edges. A vertex with no fill-in edges is called a simplicial vertex. Fast implementations of minimum degree algorithms have been developed, see e.g., AMD (Amestoy et al., 1996) with time complexity in $O(nm)$ (Heggernes et al., 2001) for an input graph G with n vertices and m edges. The minimum fill-in heuristic tends to be slower to compute but yields slightly better treewidth approximations in practice. Moreover, it will find a perfect elimination ordering (i.e., adding no fill-in edges) if it exists, thus recognizing chordal graphs and it returns the optimal treewidth in this particular case ((this can be easily established from results in Bodlaender et al., 2005).

Notice that there exists linear time $O(n + m)$ algorithms to detect chordal graphs as the Maximum Cardinality Search (MCS) greedy algorithm (Tarjan and Yannakakis, 1984) but the treewidth approximation they return is usually worse than the previous heuristic approaches.

A simple way to improve the treewidth bound found by these greedy algorithms is to break ties for the selected criterion using a second criterion, such as minimum fill-in first and then maximum degree, or to break ties at random and to iterate on the resulting randomized algorithms as done in Kask et al. (2011).

We compared the mean treewidth bound found by these four approaches (minimum degree, minimum fill-in, MCS and randomized iterative minimum fill-in) on a set of five CSP and MRF benchmarks used as combinatorial optimization problems in various solver competitions. ParityLearning is an optimization variant of the minimal disagreement parity CSP problem originally contributed to the DIMACS benchmark and used in the Minizinc challenge (Optimization Research Group, 2012). Linkage is a genetic linkage analysis benchmark (Elidan and Globerson, 2010). GeomSurf and SceneDecomp are respectively geometric surface labeling and scene decomposition problems in computer vision (Andres et al., 2013). The number of instances per problem as well as their mean characteristics are given in Table 2. Results are reported in Figure 6 (Left).The randomized iterative minimum fill-in algorithm used a maximum of $30,000$ iterations or 180 seconds (respectively $10,000$ iterations and 60 seconds for ParityLearning and Linkage), compared to a maximum of 0.37 second used by the non-iterative approaches. The minimum fill-in algorithm (using maximum degree for ties breaking) performed better than the other greedy approaches, being slightly improved by its randomized iterative version.

Problem Type/Name	Nb of instances	Mean nb of vertices	Mean nb of potential functions
CSP/ParityLearning	7	659	1246
MRF/Linkage	22	917	1560
MRF/GeomSurf-3	300	505	2140
MRF/GeomSurf-7	300	505	2140
MRF/SceneDecomp	715	183	672

Table 2: Characteristics of the five optimization problems of the benchmark. For a given problem, several instances are available, corresponding to differents number of variables (equal to the number of vertices in the underlying graph) and different numbers of potential functions.

On the same benchmark, we also compared three exact methods for the task of mode evaluation that exploit either minimum fill-in ordering or its randomized iterative version: variable elimination (ELIM), BTD (de Givry et al., 2006) and AND/OR search (Marinescu and Dechter, 2006). Elim and BTD exploit the minimum fill-in ordering while AND/OR search used its randomized iterative version. In addition, BTD and AND/OR Search exploit a tree decomposition during a Depth First Branch and Bound method in order to get a good trade-off between memory space and search effort. As variable elimination, they have a worst-case time complexity exponential in the treewidth. All methods were allocated a maximum of 1 hour and 4 GB of RAM on an AMD Operon 6176 at 2.3 GHz. The results, as reported in Figure 6 (Right) shown that BTD was able to solve more problems than the two other methods for a fixed CPU time. However, on a given problem, the best method heavily depends on the problem category. On ParityLearning, ELIM was the fastest method, but it ran out of memory on 83% of the total set of instances, while BTD (resp. AND/OR search) used less than 1.7 GB (resp. 4GB). The randomized iterative minimum fill-in heuristic used by AND/OR search in preprocessing consumed a fixed amount of time (≈ 180 seconds, included in the CPU time measurements) larger than the cost of a simple minimum fill-in heuristic. BDT was faster than AND/OR search to solve most of the instances except on two problem categories (ParityLearning and Linkage).

To perform this comparison, we ran the followinf implementation of each method. The version of ELIM was the one implemented in the combinatorial optimization solver TOOLBAR 2.3 (options `-i -T3`, available at `mulcyber.toulouse.inra.fr/projects/toolbar`). The version of BTD was the one implemented in the combinatorial optimization solver TOULBAR2 0.9.7 (options `-B=1 -O=-3 -nopre`. Toulbar2 is available at `mulcyber.toulouse.inra.fr/projects/toulbar2`. This software won the UAI 2010 (Elidan and Globerson, 2010) and 2014 (Gogate, 2014) Inference Competitions on the MAP task. AND/OR search was the version implemented in the open-source version 1.1.2 of DAOOPT (Otten et al., 2012) (options `-y -i 35 --slsX=20 --slsT=10 --lds 1 -m 4000 -t 30000 --orderTime=180` for benchmarks from computer vision and `-y -i 25 --slsX=10 --slsT=6 --lds 1 -m 4000 -t 10000 --orderTime=60` for the other benchmarks) which won the Probabilistic Inference Challenge 2011 (Elidan and Globerson, 2011), albeit with a different closed-source version (Otten et al., 2012).

5 From Variable Elimination to Message Passing

Message passing algorithms make use of messages, which can be described as potential functions which are external to the definition of graphical models. On tree-structured graphical models message passing algorithms extend the variable elimination algorithm by efficiently computing every marginals (or modes) simultaneously, when variable elimination only computes one. On general graphical models, message passing algorithms can still be applied but either provide approximate results efficiently or have an exponential running cost.

We present how it may be conceptually interesting to view these algorithms as performing a re-parametrization of the original graphical model *i.e.*, modifications of the potentials, instead of producing external messages, which are not easy to interpret by themselves.

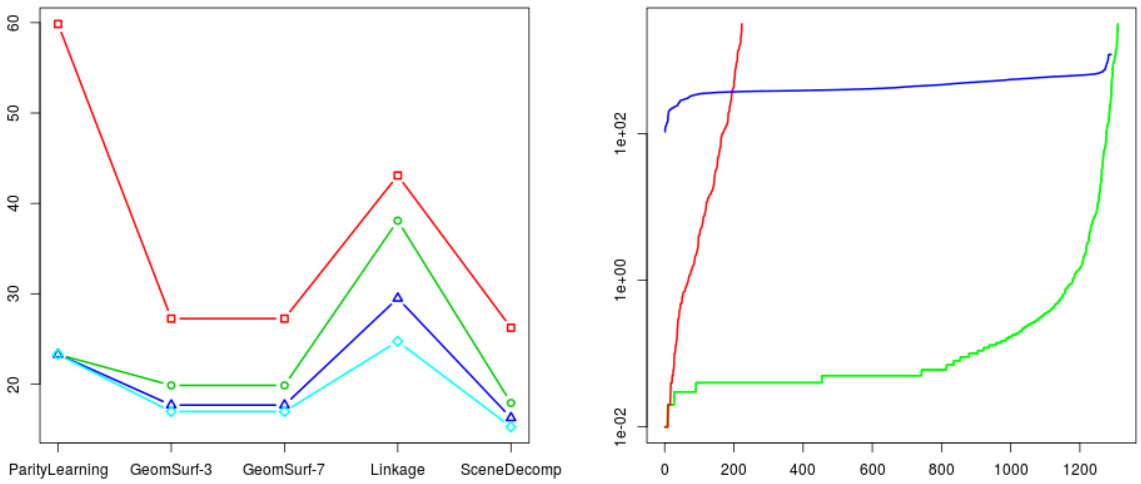

Figure 6: Left: Comparison of treewidth bounds provided by MCS (red), minimum degree (green), minimum fill-in (blue) and randomized iterative minimum fill-in (cyan) for the 5 categories of problems Right: Mode evaluation by three exact methods exploiting minimum fill-in ordering or its randomized iterative version. Number of instances solved (x-axis) within a given CPU time (log10 scale y-axis) of ELIM (red), BTD (green), and AND/OR SEARCH (blue).

5.1 Message passing and belief propagation

Message passing algorithms over trees can be described as an extension of variable elimination, where the marginals of all variables are computed in a double pass of the algorithm (instead of one variable in classical variable elimination). Instead of eliminating a leaf X_i and the potential functions involving X_i, we just mark the leaf X_i as "processed" and consider that the new potential ψ_{N_i} is a "message" sent from X_i to $X_{pa(i)}$ (the parents of X_i in the tree), denoted as $\mu_{i \to pa(i)}$. This message is a potential function over $X_{pa(i)}$ only. We can iterate this process, always applying it to a leaf in the subgraph defined by unmarked variables, handling already computed messages as unary potentials.

When only one variable remains unmarked (defining the root of the tree), the combination of all the functions on this variable (messages and possibly original potential function) will be equal to the marginal unnormalized distribution on this variable. This results directly from the fact that the operations are equivalent to variable elimination. The root of the tree defines a directed tree where the root is at the top, descendants are below and messages are flowing upwards, to the root.

To compute the marginal of another variable, one can redirect the tree using this new root. Then some subtrees will remain unchanged (in terms of direction from the root of the subtree to the leaves) in this new tree and the messages in these subtrees do not need to be recomputed.

It turns out that in a tree, one can organize all these computations cleverly so that only two messages are computed for each edge, one for each possible direction of the edge.

Formally, in the *Sum-product* algorithm over a tree (V, E), messages $\mu_{i\to j}$ are defined for each edge $(i, j) \in E$ (there are $2|E|$ such messages, one for each edge direction) in a *leaves-to-root-to-leaves* order. Messages $\mu_{i\to j}$ are functions of x_j, which are computed iteratively, by the following algorithm:

1. First, messages leaving the leaves of the tree are initialized: $\forall i \in V$, where i is a leaf of the tree,
$$\forall j, \text{ s.t. } (i, j) \in E, \forall (x_i, x_j) \in \Lambda_i \times \Lambda_j, \mu_{i\to j}(x_j) \leftarrow 1$$
Mark all leaves as processed.

2. Then, messages are sent upward through all edges. Message updates are performed iteratively, from marked nodes i to their only unmarked neighbor j through edge $(i, j) \in E$. Message updates take the following form:
$$\forall x_j \in \Lambda_j, \mu_{i\to j}(x_j) \leftarrow \frac{1}{K} \sum_{x'_i} \psi_{ij}(x'_i, x_j)\psi_i(x'_i) \prod_{k\neq j,(k,i)\in E} \mu_{k\to i}(x'_i), \tag{7}$$
where $K = \sum_{x_j} \sum_{x'_i} \psi_{ij}(x'_i, x_j)\psi_i(x'_i) \prod_{k\neq j,(k,i)\in E} \mu_{k\to i}(x'_i)$.
Mark node j as *processed*. See Figure 7 for an illustration.

3. It remains to send the latter messages downward (from root to leaves). This second phase of message updates takes the following form:
   - *U*nmark root node.
   - While there remains a marked node, send update (7) from an unmarked node to one of its marked neighbors, unmark the corresponding neighbor.

4. After the two above steps, messages have been transmitted through all edges in both directions. Finally, marginal distributions over variables and pairs of variables (linked by an edge) are computed as follows:
$$p_i(x_i) \leftarrow \tfrac{1}{K_1}\psi_i(x_i) \prod_{j,(j,i)\in E} \mu_{j\to i}(x_i), \forall x_i \in \Lambda_i,$$
$$p_{ij}(x_i, x_j) \leftarrow \tfrac{1}{K_2}\psi_{ij}(x_i, x_j) \prod_{k\neq j,(k,i)\in E} \mu_{k\to i}(x_i) \prod_{l\neq i,(l,j)\in E} \mu_{l\to j}(x_j).$$
K_1 and K_2 are suitable normalizing constants.

When the graph of the original graphical model is not a tree, the two-pass message passing algorithm can no more be applied. Still, for general graphical models, this message passing approach can be generalized in two different ways.

- One can compute a tree decomposition, as previously shown. Message passing can then be applied on the resulting cluster tree, handling each cluster as a cross-product of variables following a block-by-block approach. This yields an exact algorithm, for which computations can be expensive (exponential in the treewidth) and space intensive (exponential in the separator size). A typical example of such algorithm is the algebraic exact message passing algorithm of Shafer and Shenoy (1988); Shenoy and Shafer (1990).

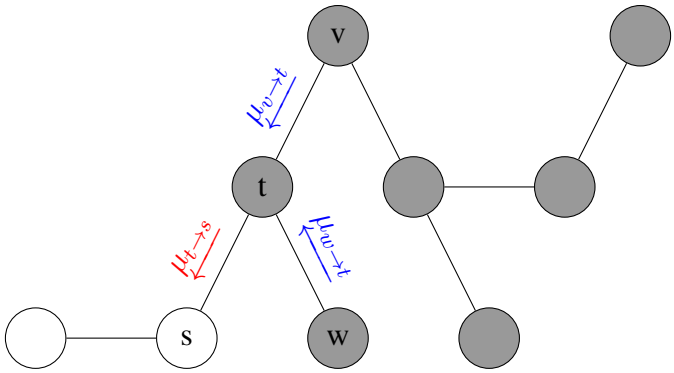

Figure 7: Example of message update on a tree. In this example, nodes t, v and w are marked, while node s is still unmarked. $\mu_{t\to s}$ is a function of all the incoming messages to node t, except $\mu_{s\to t}$.

- Alternatively, the Loopy Belief Propagation algorithm (Frey and MacKay, 1998) is another extension of Message Passing in which messages updates are repeated, in arbitrary order through all edges (possibly many times through each edge), until a termination condition is met. The algorithm returns approximations of the marginal probabilities (over variables and pairs of variables). The quality of the approximation and the convergence to steady state messages are not guaranteed (hence, the importance of the termination condition). However, it has been observed that LBP often provides good estimates of the marginals, in practice. A deeper analysis of message-passing algorithms will be provided in Section 6.

We have described above the Sum-product algorithm. Max-product, Max-sum algorithms can be equivalently defined, for exact computation or approximation of the $\max$-marginal of a joint distribution or its logarithm. In algebraic language, updates like defined in formula (7) take the general form:

$$\forall x_j \in \Lambda_j, \mu_{i\to j}(x_j) = \bigoplus_{x'_i} \psi_{ij}(x'_i, x_j)\psi_i(x'_i) \bigodot_{k\neq j,(k,i)\in E} \mu_{k\to i}(x'_i).$$

As for sum-product, the resulting algorithm computes exact $\oplus$-marginals on a tree-structured graphical model from which the mode of the distribution can be computed while on general graphical models, it provides only approximations.

5.2 Message Passing and Re-parametrization

It is possible to use message passing on trees as a re-parametrization technique. Instead of computing external messages, message passing can reformulate the original tree-structured graphical model in a new equivalent tree-structured graphical model. By "equivalent" we mean that the resulting tree defines exactly the same joint distribution as the original graphical model. In the re-parameterized problem, information of interest (marginals) can be directly read in the potential functions (Koller and Friedman, 2009).

The idea behind re-parametrization is conceptually very simple: when a message $\mu_{i\to j}$ is computed, instead of keeping it as a message, it is possible to multiply any potential function involving X_j by $\mu_{i\to j}$, using $\odot$. To preserve the joint distribution defined by the graphical model, we need to divide another potential function involving X_j by the same message $\mu_{i\to j}$ using the inverse of $\odot$.[1]

One possibility is to incorporate the messages in the binary potentials: we replace ψ_{ij} by $\psi_{ij} \odot \mu_{i\to j} \odot \mu_{j\to i}$ while ψ_i is divided by $\mu_{j\to i}$ and ψ_j is divided by $\mu_{i\to j}$. In this case, each pairwise potential ψ_{ij} can be shown to be equal to the marginal of the joint potential on $\{X_i, X_j\}$.

The resulting tree-structured MRF is said to be *calibrated* to emphasize the fact that all pairs of binary potentials sharing a common variable agree on their marginals:

$$\bigoplus_{X_j} \psi_{ij} = \bigoplus_{X_k} \psi_{ik}$$

The main advantage of a calibrated re-parametrization is that it can be used instead for the original model for any further processing. This is useful in the context of incremental updates, where new evidence is introduced incrementally and each recalibration is simpler than a new calibration (Koller and Friedman, 2009).

Message passing based re-parameterizations can be generalized to cyclic graphs. If an exact approach using tree decompositions is followed, messages may have a size exponential in the intersection of pairs of clusters and the re-parametrization will create new potentials of this size. If these messages are multiplied inside the clusters, each resulting cluster will be the marginal of the joint distribution on the cluster variables. The tree-decomposition is calibrated and any two intersecting clusters agree on their marginals. This is exploited in the Lauritzen-Spiegelhalter and Jensen sum-product-divide algorithms (Lauritzen and Spiegelhalter, 1988; Jensen et al., 1990). In this context, besides incremental updates, a calibrated tree decomposition allows also to locally compute exact marginals for any set of variables in the same cluster.

If a local "loopy" approach is used instead, re-parameterizations do not change scopes but provide a re-parameterized model where estimates of the marginals can be directly read. For MAP, such re-parameterizations can follow clever update rules to provide convergent re-parameterizations maximizing a well defined criterion. Typical examples of this schema are the sequential version of the tree reweighted algorithm (TRWS, Kolmogorov, 2006), or the Max Product Linear Programming algorithm (MPLP, Globerson and Jaakkola, 2008) which try to optimize a bound on the non-normalized probability of the mode. A seminal reference, published in Russian is (Schlesinger, 1976). These algorithms can be exact on graphical models with loops, provided the potential functions are all submodular (often described as the discrete version of convexity).

Deterministic graphical models : message passing algorithms have also been used in deterministic graphical models where they are known as "local consistency" enforcing or constraint propagation algorithms. A local consistency property defines the targeted calibration property and

[1]Zeros in potential can be dealt with by a proper extension of the algebraic operations, including an inverse for zero. If the algebraic structure equipped with $\odot$ is not a group but only a semi-group or monoid, suitable pseudo inverses can often be defined. See (Cooper and Schiex, 2004; Gondran and Minoux, 2008).

the enforcing algorithm allows to transform the original network into an equivalent network (defining the same joint function) that satisfies the desired calibration/local consistency property. Similar to LBP, Arc Consistency Waltz (1972); Rossi et al. (2006) is the most usual form of local consistency and is related to Unit Propagation in SAT Biere et al. (2009). Arc consistency is exact on trees and is usually incrementally maintained during an exact tree search, using re-parametrization. Because of the idempotency of logical operators, local consistencies always converge to a unique fix-point.

Local consistency properties and algorithms for the Weighted CSP are very closely related to message passing for MAP. They however are always convergent, thanks to suitable calibration properties (Schiex, 2000; Cooper and Schiex, 2004; Cooper et al., 2010) and may also solve tree structured or fully submodular problems.

6 Heuristics and approximations for inference

We mainly discussed methods for exact inference in graphical models. They are useful if an order for variable elimination with small treewidth is available. In real life applications, interaction network are seldom tree-shaped, and their treewidth can be large (e.g. a grid of pixel in image analysis) and exact methods cannot be applied anymore. However, they are starting points to derive heuristic methods for inference that can be applied to any graphical model. By heuristic method, we mean an algorithm that is (a priori) not derived from the optimization of a particular criterion, as opposed to what we will call approximation methods. Nevertheless, we shall alleviate this distinction and show that good performing heursitics can sometimes be interpreted as approximate methods. For the marginalization task, the most widespread heuristics derived from variable elimination and message passing principles is the Loopy Belief Propagation algorithm (LBP, Kschischang et al., 2001) described in Section 5.1, and numerous extensions (e.g. Generalized BP, Yedidia et al., 2005) have been proposed since then. In the last decade, a better understanding of these heuristics has been reached and they can now be reinterpreted as particular instances of variational approximation methods Wainwright and Jordan (2008). A variational approximation of a distribution p is defined as the best approximation of p in a class $\mathcal{Q}$ of tractable distributions (for inference), according to the Küllback-Leibler divergence. Depending of the application (e.g. discrete or continuous variables), several choices for $\mathcal{Q}$ have been considered. We are apparently far from variable elimination principles and treewidth issues. However, as we just emphasized, LBP can be cast in the variational framework. The treewidth of the chosen variational distribution depends on the nature of the variables: $i)$ in the case of discrete variables the treewidth is low: the class $\mathcal{Q}$ is in the majority of cases that of independent variables (mean field approximation), with associated treewidth of 0, and some works consider a class $\mathcal{Q}$ with associated treewidth of 1; $ii)$ in the case of continuous variables, the treewidth of the variational distribution is the same as in the original model: $\mathcal{Q}$ is in general a class of multivariate Gaussian distributions, for which numerous inference tools are available.

We will illustrate these remarks in Section 7. Before that in the remainder of this section, we recall the two key component for a variational approximation method: the Küllback-Leibler divergence and the choice of a class of tractable distributions. We also explain how LBP can be interpreted as a variational approximation method.

6.1 Variational approximations

The Küllback-Leibler divergence $KL(q||p) = \sum_x q(x) \log \frac{q(x)}{p(x)}$ measures the dissimilarity between two probability distributions p and q. KL is positive, and it is null if and only if p and q are equal. Let us consider now that q is constrained to belong to a family $\mathcal{Q}$ which does not include p. The solution q^* of $\arg\min_{q \in \mathcal{Q}} KL(p||q)$ is then the best approximation of p in $\mathcal{Q}$ according to divergence KL. If $\mathcal{Q}$ is a set of distributions tractable for inference, marginals, mode or normalizing constant of q^* can be used as approximations of the same quantities on p.

For example, let us consider a binary Potts model on n vertices whose joint distribution is

$$p(x) = \frac{1}{Z} \prod_i \exp\left(a_i x_i + \sum_{(i,j) \in E} b_{ij} x_i x_j\right)$$

We can derive its so called mean field approximation, corresponding to the class $\mathcal{Q}^{MF}$ of fully factorized distributions (i.e. an associated graph of treewidth equal to 0): $\mathcal{Q}^{MF} = \{q \; s.t. \; q(x) = \prod_{i \in V} q_i(x_i)\}$.

Since variables are binary $\mathcal{Q}^{MF}$ corresponds to joint distributions of independent Bernoulli variables with respective parameters $q_i = q_i(1)$, namely for all q in $\mathcal{Q}^{MF}$, $q(x) = \prod_i q_i^{x_i}(1 - q_i)^{1-x_i}$. The optimal approximation (in terms of Küllback-Leibler divergence) within this class of distributions is characterized by the set of q_i's which minimize $KL(q||p)$. Denoting E_q the expectation with respect to q, $KL(q||p) - \log Z$ is

$$\begin{aligned} & E_q \left(\sum_i \left[X_i \log q_i + (1 - X_i) \log(1 - q_i) \right] - \sum_i a_i X_i - \sum_{(i,j) \in E} b_{ij} X_i X_j \right) \\ & = \sum_i \left[q_i \log q_i + (1 - q_i) \log(1 - q_i) \right] - \sum_i a_i q_i - \sum_{(i,j) \in E} b_{ij} q_i q_j. \end{aligned}$$

This expectation has a simple form because of the specific structure of q. Minimizing it with respect to q_i gives the fixed-point relation that each optimal q_i^{MF}'s must satisfy:

$$\log \left[q_i^{MF} / (1 - q_i^{MF}) \right] = a_i + \sum_{j:(i,j) \in E} b_{ij} q_j^{MF}.$$

leading to

$$q_i^{MF} = \frac{e^{a_i + \sum_{j:(i,j) \in E} b_{ij} q_j^{MF}}}{1 + e^{a_i + \sum_{j:(i,j) \in E} b_{ij} q_j^{MF}}}.$$

q_i^{MF} is equal the conditional probability that $X_i = 1$ given that all other variables are fixed to their mean field expected values under distribution q, which explain the name of mean field approximation. Note that in general q_i is not equal to the marginal $p(X_i = 1)$.

The choice of the class $\mathcal{Q}$ is indeed a critical trade-off with opposite desirable properties: it must be large enough to guarantee a good approximation and small enough to contain only manageable distributions. We will focus in the next section on a particular choice for $\mathcal{Q}$, the Bethe class, that will enable to link the LBP heuristics to variational methods. Other choices are possible

and have been used. For instance, the Chow-Liu algorithm (Chow and Liu, 1968) computes the minimum of $KL(q||p)$ for q a distribution whose associated graph is a spanning tree of the graph of p. This amounts to computing the best approximation of p among graphical models with treewidth equal to 1. In the structured mean field approximation (Ghahramani and Jordan, 1997; Wainwright and Jordan, 2008) the distribution of a factorial Hidden Markov Model is approximated in a variational approach: the multivariate hidden state is decoupled and the variational distribution of the conditional distribution of hidden states is that of independent Markov chains (here again, the treewidth is equal to 1). Finally, an alternative to treewidth reduction is to choose the variational approximation in the class of exponential distributions This has been applied for Gaussian process classification (Kim and Ghahramani, 2006) using a multivariate Gaussian approximation of the posterior distribution of the hidden field. This relies on the use of the EP algorithm (Minka, 2001). Note that in this algorithm, $KL(p||q)$ is minimized instead of $KL(q||p)$.

6.2 LBP heuristics as a variational method

The mean field approximation is the most naive approximation among the so-called Kikuchi approximations from statistical mechanics, also known as Cluster Variational Methods (CVM, Kikuchi, 1951). Originally, they are not defined by a minimization of the Küllback-Leibler divergence, but as an approximation of the minimum of the free energy $H(q)$,

$$H(q) = -\sum_x q(x) \log \prod_{B \in \mathcal{B}} \psi_B(x_B) + \sum_x q(x) \log q(x).$$

The two problems are equivalent since $H(q)$ is equal to $KL(q||p) - \log Z$ and is minimum when $p = q$. If p and q are pairwise MRF whose associated graph $G = (V, E)$ is the same and is a tree, then $q(x) = \frac{\prod_{(i,j)\in E} q(x_i,x_j)}{\prod_{i\in V} q(x_i)^{d_i-1}}$, where $\{q(x_i, x_j)\}$ and $\{q(x_i)\}$ coherent sets of order 1 and order 2 marginals of q, and d_i is the degree of vertex i in the tree. In this particular case the Bethe free energy, defined as $H(q)$ is expressed as (see Heskes et al., 2004; Yedidia et al., 2005)

$$\begin{aligned} H(q) \;=\; & -\sum_{(i,j)\in E} \sum_{x_i,x_j} q(x_i, x_j) \log \psi(x_i, x_j) - \sum_{i\in V} \sum_{x_i} q(x_i) \log \psi(x_i) \\ & + \sum_{(i,j)\in E} \sum_{x_i,x_j} q(x_i, x_j) \log q(x_i, x_j) + \sum_{i\in V} (d_i - 1) \sum_{x_i} q(x_i) \log q(x_i) \end{aligned}$$

The Bethe approximation consists in applying to an arbitrary graphical model the same formula of the free energy than for a tree and minimizing it over the variables $\{q(x_i, x_j)\}$ and $\{q(x_i)\}$ under the constraint that they are probability distributions and that $q(x_i)$ is the marginal of $q(x_i, x_j)$. By extension the Bethe approximation can be interpreted as a variational method associated to the family $\mathcal{Q}^{Bethe}$ of unnormalized distributions that can be expressed as $q(x) = \frac{\prod_{(i,j)\in E} q(x_i,x_j)}{\prod_{i\in V} q(x_i)^{d_i-1}}$ with $\{q(x_i, x_j)\}$ and $\{q(x_i)\}$ coherent sets of order 1 and order 2 marginals.

It has been established by Yedidia et al. (2005) that the fixed points of LBP (when they exist, convergence is still not well understood, see Weiss (2000) and Mooij and Kappen (2007)) are stationary points of the problem of minimizing the Bethe free energy, or equivalently $KL(q||p)$ with q in the class $\mathcal{Q}^{Bethe}$ of distributions.

Furthermore Yedidia et al. (2005) showed that for any class of distributions $\mathcal{Q}$ corresponding to a particular CVM method, it is possible to define a generalized BP algorithm whose fixed points are stationary points of the problem of minimizing $KL(q||p)$ in $\mathcal{Q}$.

The drawback of the LBP algorithm and its extensions (Yedidia et al., 2005) it that they are not associated with any theoretical bound on the error made on the marginals approximations. Nevertheless, this algorithm is increasingly used for inference in graphical models for its good behavior in practice (Murphy et al., 1999). It is implemented in several pieces of software for inference in graphical models like libDAI (Mooij, 2010) or OpenGM2 (Andres et al., 2012).

7 Illustration with coupled HMM

We now illustrate how the the procedures we have described have been used for parameter estimation in a elaborated example: coupled HMM. Consider a set of I signals observed at times $t \in \{1, \dots T\}$ and denote O_t^i the variable corresponding to the observed signal i at time t. HMM models assume that the distribution of each O_t^i is conditional to some hidden state H_t^i, where the series $(H_t^i)_{t=1,\dots T}$ is a Markov chain. Coupled HMM further assumes that the hidden states display a correlation at each time (see Jordan, 2004; Wainwright and Jordan, 2008), resulting in the graphical model displayed in Figure 8. Such models have been considered in a series of domains such as bioinformatics (Choi et al., 2013), electroencephalogram analysis (Zhong and Ghosh, 2002) or speech recognition (Nock and Ostendorf, 2003). More complex versions are sometimes considered, assuming dependency between two times series at two consecutive time steps. For this model, the joint distribution of the hidden variables $H = (H_t^i)_{i,t}$ and observed variables $O = (O_t^i)_{i,t}$ factorizes as

$$p(h, o) \propto \left(\prod_i \prod_t \psi^M(h_{t-1}^i, h_t^i) \right) \times \left(\prod_t \psi^C(h_t) \right) \times \left(\prod_i \prod_t \psi^E(h_t^i, o_t^i) \right), \tag{8}$$

where $h_t = (h_t^i)_i$ stands the vector of all hidden variables at time t and where ψ^M encodes the Markovian dependency of the hidden variables within a series, ψ^C encodes the coupling between the hidden variables of all series at a given time and ψ^E encodes the emission of the observed signal given the corresponding hidden state. A fairly comprehensive exploration of these models can be found in (Murphy, 2002).

7.1 Exact EM algorithm for coupled HMM

Coupled HMM are examples of incomplete data models, as they involve variables (O, H) where only the variables O are observed. Maximum likelihood inference for such a model aims at finding the value of the parameter θ that maximizes the (log-)likelihood of the observed data o, that is to solve $\max_\theta \log p^\theta(o)$. The most popular algorithm to achieve this task is the EM algorithm, (Dempster et al., 1977), that can be rephrased in the following way. Observe that

$$\log p^\theta(o) = E(\log p^\theta(o, H)|o) - E(\log p^\theta(H|o)|o) = \max_q F(\theta, q),$$

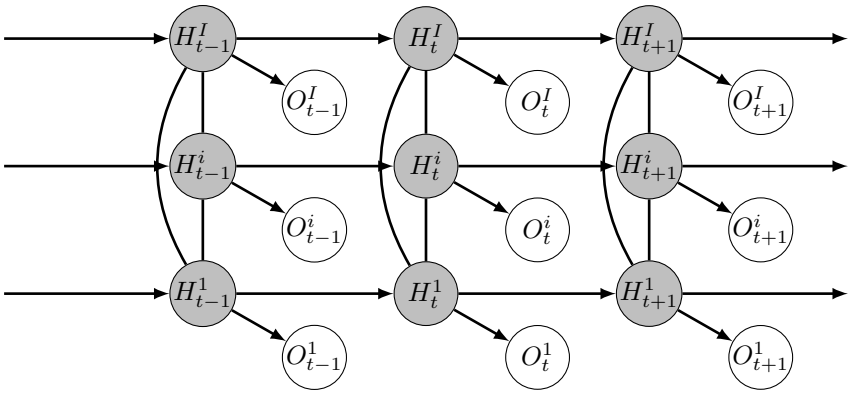

Figure 8: Graphical representation of $p(h, o)$ for a coupled HMM

where

$$F(\theta, q) = E_q(\log p^\theta(o, H)) - E_q(\log q(H)) = \log p^\theta(o) - KL(q(H)||p^\theta(H|o)),$$

and q stands for any distribution on the hidden variables H, E stands for the expectation under the true distribution p and E_q under the arbitrary distribution q. The EM algorithm consists in alternatively maximizing $F(\theta, q)$ with respect to q (E-step) and to θ (M-step). The solution of the E-step is $q(h) = p(h|o)$ since the Kullback-Leibler divergence is minimal (and null) in this case.

Exact computation of $p(h|o)$ can be performed by observing that (8) can be rewritten as

$$p(h, o) \propto \left(\prod_t \psi^{M'}(h_{t-1}, h_t) \right) \times \left(\prod_i \prod_t \psi^E(h^i_t, o^i_t) \right),$$

where $\Psi^{M'}$ encodes both the Markovian dependency and the coupling of the hidden variables within a given time step. This writing is equivalent to merging all hidden variables of a given time step and corresponds to the graphical model given in Figure 9. Denoting K the number of possible values for each hidden variables, we end up with a regular hidden Markov model with K^I possible hidden states. Both $p(h|o)$ (and its mode) can then be computed in a exact manner with either the forward-backward recursion or the Viterbi algorithm, which have the same complexity: $O(TK^{2I})$. The exact calculation can therefore be achieved provided that K^I remains small enough, but becomes intractable when the number of signals I exceeds few tens.

7.2 Approximate E step for the EM algorithm

A first approach to derive an approximate E step it to seek for a variational approximation of $p(h|o)$ assuming that $q(h)$ is restricted to a family $\mathcal{Q}$ of tractable distributions, as described in Section 6.1. This approach results in the maximization of a lower bound of the original log-likelihood. The choice of $\mathcal{Q}$ is critical and is a balance between approximation accuracy and computation efficiency. Choosing $\mathcal{Q}$ typically amounts to breaking down some dependencies in the original

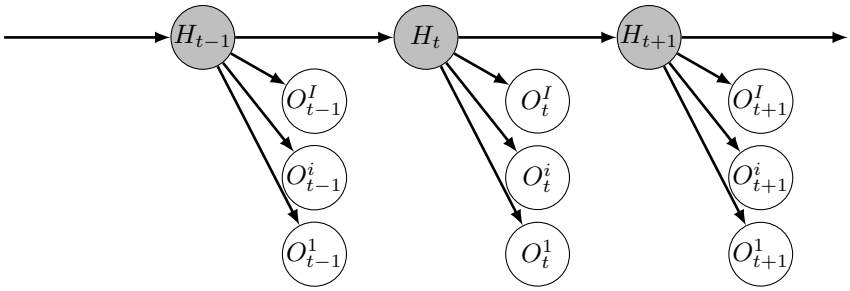

Figure 9: Graphical representation of $p(h, o)$ for a coupled HMM when merging hidden variables at each time step

distribution to end up with some tractable distribution. In the case of coupled HMM, the simplest distribution is the class of fully factorized distributions (i.e. mean field approximation), that is

$$\mathcal{Q}_0 = \{q : q(h) = \prod_i \prod_t q_{it}(h^i_t)\}.$$

Such an approximation corresponds to the graphical model of Figure 10. Intuitively, this approximation replaces the stochastic influence existing between the hidden variables by its mean value.

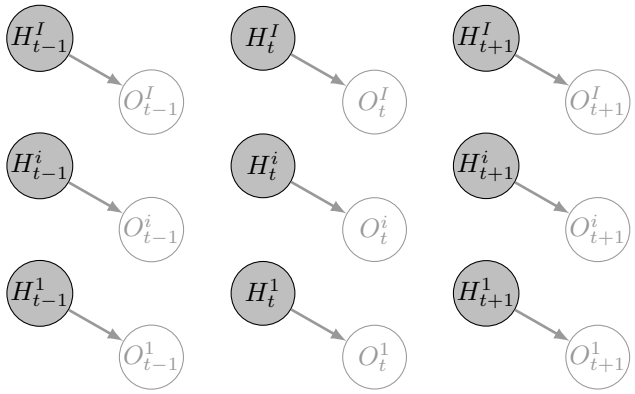

Figure 10: Graphical representation for the independent mean-field approximation of $p(h, o)$ in a coupled HMM. Observed variables are indicated in light gray since they are not part of the variational distribution which is a distribution only on the hidden variables.

As suggested in Wainwright and Jordan (2008), a less drastic approximation can be obtained using the distribution family of independent heterogeneous Markov chains:

$$\mathcal{Q}_M = \{q : q(h) = \prod_i \prod_t q_{it}(h^i_t | h^i_{t-1})\}$$

which is consistent with the graphical representation of independent HMM, as depicted in Figure 11.

An alternative to the approximate maximization of $\max_q F(\theta, q)$ consists in seeking for the maximum of an approximation $\widetilde{F}(\theta, q)$ of $F(\theta, q)$ which involves only marginals of $p(h|o)$ on

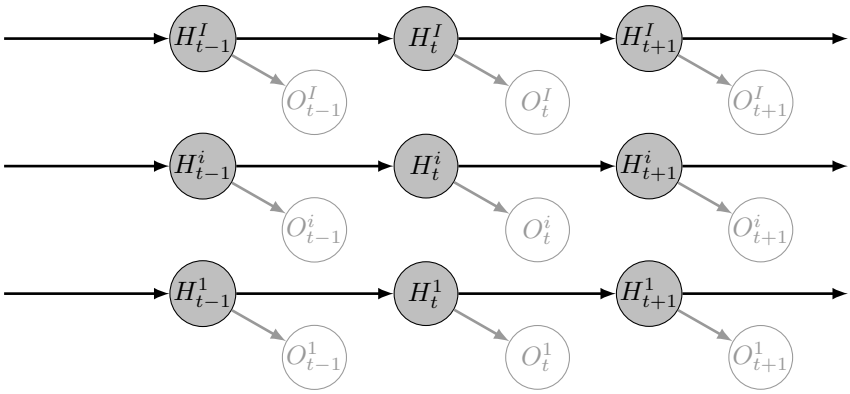

Figure 11: Graphical representation for the independent Markov approximation of $p(h, o)$ in a coupled HMM. Observed variables are indicated in light gray since they are not part of the variational distribution which is a distribution only on the hidden variables.

subsets of variables in H of limited size. Then the LBP algorithm can be used to provide an approximation of these marginals. This approach has been proposed in Heskes et al. (2004) where the authors approximated the negative entropy term $E_q(\log q(H))$ in $F(\theta, q)$ by its so-called Bethe approximation as follows (the first term in F by definition depends only on marginals of variables involved in the potential functions ψ^M, ψ^C, ψ^E).

$$\sum_i \sum_t q^M(h^i_{t-1}, h^i_t) \log q^M(h^i_{t-1}, h^i_t) + \sum_t q^C(h_t) \log q^C(h_t)$$
$$+ \sum_i \sum_t q^E(h^i_t, o^i_t) \log q^E(h^i_t, o^i_t) - \sum_i \sum_t Iq(h^i_t) \log q(h^i_t)$$

because each hidden variable H^i_t has degree $I + 1$ in the original graphical model given in Figure 8. The advantage of this approach compared to the variational approximations based on families $\mathcal{Q}_0$ and $\mathcal{Q}_M$ is that it provides an approximation of the joint conditional distribution of all hidden variables within the same time step.

8 Conclusion

This tutorial on variable elimination for exact and approximate inference is an introduction to the basic concepts of variable elimination, message passing and their links with variational methods. It introduces these fields to statisticians confronted with inference in graphical models. The main message is that exact inference should not be systematically considered as out of reach. Before looking for an efficient approximate method, a wise advice would be to know the treewidth of the graphical model. In practice this question is not easy to answer exactly but several implementations of the presented algorithms exist and provide an upper bound of the treewidth.

Obviously this tutorial is not exhaustive, since we have chosen to focus on the fundamental concepts. While many important results on treewidth and graphical models have several decades

in age, the area is still lively and we now broaden our discussion to a few recent works which tackle some challenges related to the computation of the treewidth.

Because they offer efficient algorithms, graphical models with a bounded treewidth offer an attractive target when the aim is to learn a model that best represents some given sample. In (Kumar and Bach, 2012), the problem of learning the structure of an undirected graphical model with bounded treewidth is approximated by a convex optimization problem. The resulting algorithm has a polynomial time complexity. As discussed in (Kumar and Bach, 2012), this algorithm is useful to derive tractable candidate distributions in a variational approach, enabling to go beyond the usual variational distributions with treewidth zero or 1.

For optimization (MAP), other exact techniques are offered by tree search algorithms such as Branch and Bound (Lawler and Wood, 1966), that recursively consider possible conditioning of variables. These techniques often exploit limited variable elimination processing to prevent exhaustive search, either using message-passing like algorithms (Cooper et al., 2010) to compute bounds that can be used for pruning, or by performing "on-the-fly" elimination of variables with small degree (Larrosa, 2000).

Beyond pairwise potential functions, the time needed for simple update rules of message passing becomes exponential in the size of the scope of the potential functions. However, for specific potential functions involving many (or all) variables, exact messages can be computed in reasonable time, even in the context of convergent message passing for optimization. This can be done using polytime graph optimization algorithms such as shortest path or mincost flow algorithms. Such functions are known as global potential functions (Vicente et al., 2008; Werner, 2008) in stochastic graphical models, and as global cost functions (Lee and Leung, 2009; Allouche et al., 2012; Lee and Leung, 2012) in deterministic Cost Function Networks.

Different problems appears with continuous variables, where counting requires integration of functions. Here again, for specific families of distributions, exact (analytic) computations can be obtained for distributions with conjugate distributions. For message passing, several solutions have been proposed. For instance, a recent message passing scheme proposed by Noorshams and Wainwright (2013) relies on the combination of orthogonal series approximation of the messages, and the use of stochastic updates. We refer the reader to references in (Noorshams and Wainwright, 2013) for a state-of-the-art of alternative methods dealing with continuous variables message passing.

Finally, we have excluded Monte-Carlo methods from the scope of our review. But recent sampling algorithms have been proposed that use exact optimization algorithms to sample points with high probability in the context of estimating the partition function. Additional control in the sampling method is needed to avoid biased estimations: this may be hashing functions enforcing a fair sampling (Ermon et al., 2014) or randomly perturbed potential functions using a suitable noise distribution (Hazan et al., 2013).

We hope this review will enable more cross-fertilizations of this sort, combining statistics and computer science, stochastic and deterministic algorithms for inference in graphical models.

References

D. Allouche, C. Bessiere, P. Boizumault, S. de Givry, P. Gutierrez, S. Loudni, J.P. Metivier, and T. Schiex. Decomposing Global Cost Functions. In *Proceedings of AAAI'12*, pages 407–413, 2012.

P. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

E. Amir. Approximation algorithms for treewidth. *Algorithmica*, 56:448–479, 2010.

B. Andres, Beier T., and J. H. Kappes. OpenGM: A C++ library for discrete graphical models. *ArXiv e-prints*, 2012. URL `http://hci.iwr.uni-heidelberg.de/opengm2/`.

Bjoern Andres, Thorsten Beier, and Joerg H. Kappes. Open gm benchmark - cvpr'2013 section. See `http://hci.iwr.uni-heidelberg.de/opengm2/?l0=benchmark`, 2013.

S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability — a survey. *BIT*, 25:2–23, 1985.

S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8:277–284, 1987.

P. Austrin, T. Pitassi, and Y. Wu. Inapproximability of treewidth, one-shot pebbling, and related layout problems. In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX),*, pages 13–24, Boston, USA, 2012.

U. Bertelé and F. Brioshi. *Nonserial Dynamic Programming*. Academic Press, 1972.

Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. Ios press, 2009.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

S. Bistarelli, U. Montanari, and F. Rossi. Semiring based constraint solving and optimization. *Journal of the ACM*, 44(2):201–236, 1997.

H. L. Bodlaender. A tourist guide through treewidth. *Developments in Theoretical Computer Science*, 1, 1994.

H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

H. L. Bodlaender and A. M. C. A. Koster. Treewidth computations I. upper bounds. *Information and Computation*, 208(3):259–275, 2010.

H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, 18:238–255, 1995.

H. L. Bodlaender, A. Koster, and F. van den Eijkhof. Preprocessing rules for triangulation of probabilistic networks. *Computational Intelligence*, 21(3):286–305, 2005.

H. L. Bodlaender, P. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. In *IEEE Symposium on Foundations of Computer Science*, pages 499–508, 2013.

Hans L Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical computer science*, 209(1):1–45, 1998.

G. Casella and E. I. George. Explaining the Gibbs sampler. *The American Statistician*, 46(3): 167–174, 1992.

H. Choi, D. Fermin, A. Nesvizhskii, D. Ghosh, and Z. Qin. Sparsely correlated hidden Markov models with application to genome-wide location studies. *Bioinformatics*, 29(5):533–541, 2013.

C. K. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

S.A. Cook. The complexity of theorem proving procedures. In *3rd ACM symp. on theory of computing*, pages 151–158, 1971.

M C. Cooper. Cyclic consistency: a local reduction operation for binary valued constraints. *Artificial Intelligence*, 155(1-2):69–92, 2004.

M C. Cooper and T. Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154(1-2): 199–227, 2004.

M. C. Cooper, S. de Givry, M. Sanchez, T. Schiex, M. Zytnicki, and T. Werner. Soft arc consistency revisited. *Artificial Intelligence*, 174:449–478, 2010.

M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):210–215, 1960.

S. de Givry, T. Schiex, and G. Verfaillie. Exploiting Tree Decomposition and Soft Local Consistency in Weighted CSP. In *Proceedings of the National Conference on Artificial Intelligence, AAAI-2006*, pages 22–27, 2006.

R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113 (1–2):41–85, 1999.

R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. In L. Kanal and V. Kumar, editors, *Search in Artificial Intelligence*, chapter 11, pages 370–425. Springer-Verlag, 1988.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

R. J. Duffin. Topology of series-parallel networks. *Journal of Mathematical Analysis and Application*, 10(2):303–313, 1965.

G. Elidan and A. Globerson. UAI inference challenge 2010. See www.cs.huji.ac.il/project/UAI10, 2010.

G. Elidan and A. Globerson. The probabilistic inference challenge. See http://www.cs.huji.ac.il/project/PASCAL/index.php, 2011.

S. Ermon, C. Gomes, A. Sabharwal, and B. Selman. Low-density parity constraints for hashing-based discrete integration. In *Proceedings of the 31st International Conference on Machine Learning*, pages 271–279, 2014.

F. V. Fomin and Y. Villanger. Treewidth computation and extremal combinatorics. *Combinatorica*, 32(3):289–308, 2012.

J. Fourier. *Mémoires de l'Académie des sciences de l'Institut de France 7*, chapter Histoire de l'Académie, partie mathématique (1824). Gauthier-Villars., 1827.

B. Frey and D. MacKay. A revolution: Belief propagation in graphs with cycles. In *Advances in Neural Information Processing Systems*, pages 479–485. MIT Press, 1998.

N. Friel, A. N. Pettitt, R. Reeves, and E. Wit. Bayesian inference in hidden Markov random fields for binary data defined on large lattices. *Journal of Computational and Graphical Statistics*, 18: 243–261, 2009.

D. Fulkerson and O. Gross. Incidence matrices and interval graphs. *Pacific journal of mathematics*, 15(3):835–855, 1965.

Z. Ghahramani and M. Jordan. Factorial hidden Markov models. *Machine learning*, 29(2-3): 245–273, 1997.

A; Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In *Advances in Neural Information Processing Systems*, pages 553–560, 2008.

V. Gogate. UAI 2014 inference competition. See www.hlt.utdallas.edu/~vgogate/uai14-competition, 2014.

M. Gondran and M. Minoux. *Graphs, Dioids and Semirings*, volume 41 of *Operations Research/Computer Science Interfaces Series*. Springer, 2008.

D. J. Gordon, N. J.and Salmond and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings on Radar and Signal Processing*, 140(2):107–113, 1993.

T. Hazan, S. Maji, and T. Jaakkola. On sampling from the Gibbs distribution with random maximum a-posteriori perturbations. In *Advances in Neural Information Processing Systems*, pages 1268–1276, 2013.

P. Heggernes, S. Eisenstat, G. Kumfert, and A. Pothen. The computational complexity of the minimum degree algorithm. In *14th Norwegian Computer Science Conference*, Troms, Norway, 2001.

T. Heskes, O. Zoeter, and W. Wiegerinck. Approximate expectation maximization. *Advances in Neural Information Processing Systems*, 16:353–360, 2004.

F. Jensen, K. Olesen, and S. Andersen. An algebra of bayesian belief universes for knowledge-based systems. *Networks*, 20(5):637–659, 1990.

F. V. Jensen and T. D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer Publishing Company, Incorporated, 2nd edition, 2007.

M. Jordan. Graphical models. *Statistical Science*, pages 140–155, 2004.

K. Kask, A. Gelfand, L. Otten, and R. Dechter. Pushing the power of stochastic greedy ordering schemes for inference in graphical models. In *AAAI*, 2011.

R. Kikuchi. A theory of cooperative phenomena. *Physical Review*, 81(6):988–1003, 1951.

H.C. Kim and Z. Ghahramani. Bayesian gaussian process classification with the EM-EP algorithm. *IEEE Transactions on Pattern Analyis and Machine Intelligence*, 28(12):1948–1959, 2006.

J. Kohlas. *Information algebras: Generic structures for inference*. Springer Science & Business Media, 2003.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

F. R. Kschischang, B. J. Frey, and H-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498 –519, 2001.

K. S. Sesh Kumar and F. Bach. Convex relaxations for learning bounded treewidth decomposable graphs. In *Proceedings of the International Conference on Machine Learning (ICML)*, Atlanta, United States, 2012.

J. Lagergreen. Efficient parallel algorithms for graphs of bounded treewidth. *Journal of Algorithms*, 20:20–44, 1996.

J. Larrosa. Boosting search with variable elimination. In *Principles and Practice of Constraint Programming - CP 2000*, volume 1894 of *LNCS*, pages 291–305, Singapore, September 2000.

S. L. Lauritzen. *Graphical Models*. Clarendon Press, 1996.

S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society – Series B*, 50:157–224, 1988.

E. Lawler and D. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4): 699–719, 1966.

J. Lee and K. L. Leung. Towards efficient consistency enforcement for global constraints in weighted constraint satisfaction. In *International Conference on Artificial Intelligence*, volume 9, pages 559–565, 2009.

J. H. M. Lee and K. L. Leung. Consistency Techniques for Global Cost Functions in Weighted Constraint Satisfaction. *Journal of Artificial Intelligence Research*, 43:257–292, 2012.

S. Z. Li. *Markov random field modeling in image analysis*. Springer-Verlag, 2001.

J. W. H. Liu. The multifrontal method for sparse matrix solution: Theory and practice. *SIAM Review*, 34:82–109, 1992. First papers on the multifrontal technique go back to 1983.

L. Lovász. Graph minor theory. *Bulletin of the American Mathematical Society*, 43:75–86, 2005.

R. Marinescu and R. Dechter. Memory intensive branch-and-bound search for graphical models. In *proceedings of the National Conference on Artificial Intelligence, AAAI-2006*, pages 1200–1205, 2006.

T. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT, 2001.

J. M. Mooij. libDAI: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11:2169–2173, August 2010. URL `https://staff.fnwi.uva.nl/j.m.mooij/libDAI/`.

J. M. Mooij and H. J. Kappen. Sufficient conditions for convergence of the sum-product algorithm. *IEEE Transactions on Information Theory*, 53(12):4422–4437, 2007.

K. Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

K. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the 15th conference on Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

H. Nock and M. Ostendorf. Parameter reduction schemes for loosely coupled HMMs. *Computer Speech & Language*, 17(2):233–262, 2003.

N. Noorshams and M. J. Wainwright. Belief propagation for continuous state spaces: stochastic message-passing with quantitative guarantees. *Journal of Machine Learning Research*, 14(1): 2799–2835, 2013.

NICTA Optimization Research Group. Minizinc challenge 2012. See `http://www.minizinc.org/challenge2012/challenge.html`, 2012.

L. Otten, A. Ihler, K. Kask, and R. Dechter. Winning the PASCAL 2011 MAP challenge with enhanced AND/OR branch-and-bound. In *NIPS DISCML Workshop*, Lake Tahoe, USA, 2012.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems, Networks of Plausible Inference*. Morgan Kaufmann, Palo Alto, 1988.

L. Perkovic and B. Reed. An improved algorithm for finding tree decompositions of small treewidth. *International Journal of Foundations of Computer Science*, 11:365–371, 2000.

C. Pralet, G. Verfaillie, and T. Schiex. An algebraic graphical model for decision with uncertainties, feasibilities, and utilities. *Journal of Artificial Intelligence Research*, pages 421–489, 2007.

L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

R. Reeves and A. N. Pettitt. Efficient recursions for general factorisable models. *Biometrika*, 91: 751–757, 2004.

C.P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer- Verlag, New York, 2004.

N. Robertson and P. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

T. Schiex. Arc consistency for soft constraints. In *Principles and Practice of Constraint Programming - CP 2000*, volume 1894 of *LNCS*, pages 411–424, Singapore, September 2000.

T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: hard and easy problems. In *Proc. of the 14th IJCAI*, pages 631–637, Montréal, Canada, August 1995.

M.I. Schlesinger. Sintaksicheskiy analiz dvumernykh zritelnikh signalov v usloviyakh pomekh (Syntactic analysis of two-dimensional visual signals in noisy conditions). *Kibernetika*, 4:113–130, 1976.

G. Shafer and P. Shenoy. Local computations in hyper-trees. Working paper 201, School of business, University of Kansas, 1988.

P. Shenoy and G. Shafer. Axioms for probability and belief-function propagation. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*, pages 169–198, Cambridge, USA, 1990.

R. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM Journal of Computing*, 13(3):566–579, 1984.

L. Vandenberghe and M. S. Andersen. Chordal graphs and semidefinite optimization. *Foundations and Trends in Optimization*, 1(4):241–433, 2014.

S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *Computer Vision and Pattern Recognition,CVPR 2008*, pages 1–8, Alaska, USA, 2008.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

D. L. Waltz. Generating semantic descriptions from drawings of scenes with shadows. Technical Report AI271, M.I.T., Cambridge MA, 1972.

Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12(1), 2000.

T. Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (map-mrf). In *Computer Vision and Pattern Recognition, CVPR 2008*, pages 1–8, Alaska, USA, 2008.

J. Yedidia, W. Freeman, and Y. Weiss. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

S. Zhong and J. Ghosh. HMMs and coupled HMMs for multi-channel EEG classification. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 2, pages 1254–1159, Honolulu, Hawai, 2002.